REPORT RESUMES

ED 013 448 24 AL 000 668

AN INVESTIGATION OF THE USE OF DIGITAL COMPUTERS IN THE CREATION OF PROGRAMMED LANGUAGE INSTRUCTION, WITH PARTICULAR REFERENCE TO STANDARD UKRAINIAN AND AMERICAN ENGLISH.

BY- VAN CAMPEN, JOSEPH A.

REPORT NUMBER BR-5-8212 PUB DATE APR 67
CONTRACT OEC-5-10-115
EDRS PRICE MF-$0.75 HC-$5.40 135P.

DESCRIPTORS- *COMPUTATIONAL LINGUISTICS, *UKRAINIAN, ENGLISH, *COMPUTER ASSISTED INSTRUCTION, *INFORMATION PROCESSING, SYNTAX, MORPHOLOGY (LANGUAGES), PHONOLOGY, DISTINCTIVE FEATURES, DIGITAL COMPUTERS, PROGRAMED TEXTS, PROGRAMING PROBLEMS, CONTRASTIVE LINGUISTICS, IBM 7094,

BEFORE ESTABLISHING HOW A DIGITAL COMPUTER COULD BE USED FOR DESCRIBING BASIC CONTRASTS BETWEEN AMERICAN ENGLISH AND STANDARD UKRAINIAN, IT WAS NECESSARY TO DEVELOP A SET OF SYMBOLS CORRESPONDING TO CERTAIN ELEMENTARY DESCRIPTIVE UNITS OF GRAPHEMICS, PHONOLOGY, MORPHOLOGY, AND SYNTAX AND TO APPLY THESE SYMBOLS TO ENCODING FOR THE IBM 7094. THE RESEARCH WAS MAINLY UNDERTAKEN TO DETERMINE HOW UNFAMILIAR INFORMATION ON A "TARGET LANGUAGE" COULD BE ORDERED IN THE STEP-BY-STEP PROCEDURE CHARACTERISTIC OF PROGRAMED INSTRUCTION. DESCRIBING THE PHONOLOGY WAS FURTHER COMPLICATED BECAUSE NEITHER THE DISTINCTIVE FEATURE APPROACH NOR THE ARTICULATORY/ACOUSTIC APPROACH WAS COMPLETELY ADEQUATE, BOTH BEING AMBIGUOUS. ANOTHER DIFFICULTY AROSE FROM THE FACT THAT THE DESCRIPTIONS OF ENGLISH HAD BEEN WRITTEN BY BRITISH OR AMERICAN LINGUISTS, WHILE THE DESCRIPTIONS OF UKRAINIAN REFLECTED EAST EUROPEAN OR SOVIET LINGUISTIC THOUGHT. THE FINAL OBJECTIVE WAS TO CREATE ONE OR MORE OUTLINES FOR PROGRAMED INSTRUCTION IN UKRAINIAN FOR ENGLISH SPEAKERS. THIS PROVED UNREALISTIC BECAUSE THE COMPUTER WAS NOT CAPABLE OF FULLY ASSESSING THE DIFFERENCES AND SIMILARITIES BETWEEN LANGUAGES. (FB)

BR 5-8212
PA-24

ED013448

Final Report on Contract Number OE 5-10-115

Project Title: An Investigation of the Use of Digital
Computers in the Creation of Programmed
Language Instruction, with Particular
Reference to Standard Ukrainian and
American English.

Principal Investigator: Joseph A. Van Campen

Date Submitted: April 1967

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

AL 000 668

0. Introductory remarks.

In order to facilitate the comprehension of this report by the layman we reproduce below the abstract contained in the original proposal. The section of this abstract entitled 'Procedures' will serve as the basis of reference throughout the entire report.

Objectives: 1) To illustrate the extent to which a digital computer can be used to determine a) what orthographic, phonological, morphological, or syntactic features of one language are unfamiliar to speakers of another; b) how such unfamiliar information on a "target" language can be introduced in the ordered, step-by-step fashion characteristic of programmed instruction. 2) To set the stage for research on the use of a digital computer in selecting a sequence of presentation and reinforcement frames which will allow the introduction and reinforcement of a given information sequence with a maximum of reinforcement and a minimum of monotony.

Procedures:

1) To develop a set of symbols corresponding to certain elementary descriptive units of graphemics, phonology, morphology, and syntax and to employ these symbols in encoding for use by the IBM 7094 computer a description of the orthography, phonology, morphology and syntax of modern standard Ukrainian and American English.

2) To encode for the computer a basic lexicon of some 5,000 items in the target language (Ukrainian), including in the lexical entries any information required for the correct application thereto of the rules in the grammar of that language.

3) To create a program enabling the computer to compare the descriptions of English and Ukrainian and to distinguish symbols or rules found in both descriptions from symbols or rules found only in the description of the target language.

4) To create a program enabling the computer to order the introduction of information on Ukrainian not familiar to speakers of English by making decisions as to the desirability of introducing a given lexical item or rule at a given point in the program.

5) To apply the programs mentioned in 3) and 4) to the computer-encoded descriptions and lexicon to create one or more outlines for programmed instruction in Ukrainian for speakers of English.

(If such a step is felt desirable, it should be possible at a later date to use part of such a program outline as the basis for a programmed introductory course in Ukrainian for speakers of English and to test this course in the Department of Slavic Languages and Literatures at Harvard University. These steps, however, are not within the scope of the present proposal.)

1. Development of symbols and encoding of description of orthography, phonology, morphology and syntax.

1.1 General remarks.

Despite the apparently simple nature of this section of the project, it proved to offer enormous difficulties. Indeed, at least within the limits of a project of this size, many serious problems remain essentially unsolvable, although materials have been developed which show what a true solution might look like if one could be attained.

1.2 Phonology.

This area is one on which a vast amount of previous research has been done. At first glance, it would seem that one would merely have to combine the achievements of the distinctive-feature approach with those of articulatory and/or acoustic phonetics to achieve this goal. In reality, neither of these approaches proved satisfactory: the distinctive-feature approach was found to be ambiguous and potentially misleading; the phonetic approach suffered because of 1) the ambiguity of and differences in its terminology 2) the varying amount of detail provided by different discriptions.

1.21 Distinctive-feature approach.

1.211 Ambiguity.

Despite the fact that the distinctive-feature approach was designed to allow linguists to determine the maximally economical approach to the description of the phonology of a given language, it is extremely difficult if nor impossible to determine just which set of descriptive statements meets this criterion. This difficulty has its origin in the fact that distinctive-feature descriptions consist not only of feature specifications but also of phonological rules. Consequently, a simple enumeration of feature specifications does not give a complete picture of the description's economy, since a smaller number of specifications may entail an extremely complicated set of rules, while a somewhat larger set of specifications may allow the use of a much simpler set of rules.

Moreover, quite different descriptions can contain one and the same number of specifications. For example, it is possible to view Russian /f/ as either distinctly strident or distinctly continuous. In either case it is set off from /p/, which is either mellow or interrupted, respectively, and hence the "structural economy" involved is the same. However, it is clear that from the point of view of a feature-by-feature comparison with the phonology of another language, the difference is very large indeed.

1.212 Misleading nature.

Quite aside from the fact that it does not always offer unambiguous solutions, the distinctive-feature approach has yet another drawback: since the inventory of distinctive features reflects only those features which may be opposed to one another within the framework of one and the same language, it fails to account for many significant phonetic differences. Thus, for instance, the distinctive feature "flatness" includes such phonetic features as "lip rounding", "pharyngealization", and "retroflex articulation"--features which can hardly be substituted freely one for the other within the framework of a single language. Again, the distinctive feature "tense" can be used of both aspirated and unaspirated stops, although it is clear that in many languages with tense unaspirated stops [e.g. French] the use of aspirated stops is unacceptable.

1.22 Phonetic approach.

The above-outlined effects of the distinctive-feature approach led the principal investigator to place a greater emphasis than had been originally anticipated on articulatory phonetics. However, even here a truly objective solution proved impossible of attainment within the framework of the present project.

1.221. Terminological difficulties.

The most striking difficulties in this area arose from the fact that the descriptions of English used had been written in English by American or British linguists, while the sources for Ukrainian were in Ukrainian or Polish and represented East European or Soviet approaches. However, it is quite possible to find differences in usage within the material on either language. Thus, some American writers fail to distinguish between the post-dental and the alveolar articulation of stops. Again, while Zilynskij distinguishes at the very least nine different degrees of "openness" for Ukrainian vowels, more recent writers use only three basic degrees supplemented by two intermediary ones. Such terminological variation can have an important impact on the results of the comparison of two phonologies. Thus, if we take our information on English /t/ from an author in whose terminology alveolar stops are "made behind the upper teeth" the coding for the English sound will be identical with that for (postdental) Ukranian /t/ as regards place of articulation. If, on the other hand, we take our information on English from an author who sharply distinguishes the two classes, the coding will be different.

1.222 Variability of descriptions with regard to detail.

Certain aspects of the terminological differences (e.g. Zylinskij's use of nine degrees of "openness") discussed above can themselves be viewed as differences in detail. However, even in cases of relatively similar terminological usage it is possible to find great variations in the amount of information presented. Thus, both Daniel Jones and the Ukrainian sources make fairly similar use of the terms "tense" and "lax". However, Jones provides information regarding this feature for English consonants, while in the Ukrainian sources it is mentioned only in connection with vowels. Another striking example is the wealth of material provided by Jones on the partial loss of voicing by English stops and spirants in, for example, initial or final position. Many American sources simply ignore the partially voiceless variants. Depending on the source chosen, the inventory of English sounds (and, in particular, of sounds differing from Ukrainian sounds) would include or lack a dozen or more variants).

1.23 Dialect Variations.

Quite aside from the problems mentioned under 1.21 and 1.22, the satisfactory treatment of phonological differences was rendered even more difficult by the existence within both standard English and standard Ukrainian of significant dialectal variations. The most troublesome problem was entailed by the fact that the best description of a variant of English--that by Daniel Jones--deals with a variety of British English. However, even within American speech the number and type of vowel nuclei varies so widely that vowels in a foreign language which are alien to one speaker may be quite familiar to another. In order not to sacrifice all of the detail of Jones' description it was decided to include those of his statements which applied also to the principal investigator's dialect of American English. The choice of vowel nuclei was also based to a large extent on the speech of the principal investigator. The choice of Ukrainian material was based on the Eastern Ukrainian variant of the standard language.

1.24 A tentative coding system for phonological information.

The difficulties listed under 1.21 and 1.22 resulted in the establishment of a coding system for phonological information which, it is hoped, avoids the most serious disadvantages of both the distinctive-feature and the phonetic approach. On the one hand, we distinguish significantly different phonetic features which represent variants of one and the same distinctive feature; on the other, we have retained certain distinctive features (e.g. strident, compact and diffuse) which permit the handling of significant differences in a more efficient way than would a purely phonetic approach.

1.241 Choice of units to be coded.

The phonological coding is given only for individual sounds (allophones). No attempt has been made to provide coding for such units as phonemes or morphonemes, since any such attempt involves a return to the ambiguity and misleading selectivity of the distinctive feature approach. The sole purpose of the phonological information is to allow the computer to decide whether a sound which regularly occurs in the Ukrainian speech being described also regularly occurs in the English speech being described. The conditions determining its occurrence are treated elsewhere (cf. 4.2).

1.242 Illustration of the coding of phonological information on the basis of standard English and Ukrainian.

Appendix 1 , shows the results of the application of our coding to the allophones of the varieties of standard Ukrainian and standard English discussed under 1.3. The material is given as it might appear on a set of IBM cards.

The first part of each card (columns 1 through 10) is the label and consists of 1) an initial character (U for Ukrainian, E for English) identifying the language involved (note that the initial character is omitted for all allophones except UP1 on page S1 and EK1 on page S5), and 2) an allophone identification set consisting of one or more letters followed by a numeral. The coding of phonological information begins in column 11 and continues through column 49. However, additional information might be coded in columns 50 through 72. In the example given the only coding possibililies in any given column are a one or a blank.

The symbols above the individual columns represent, in the order of their occurrence, the following features: C--consonantal, V--vocalic, CP--compact, **D--diffuse**, G--grave, STR--strident, CT--continuous, N--nasal, VD--voiced, TNS--tense, SHP--sharp, FLT--flat, CHK--checked, ACC--accented, SYL--syllabic, LNG--long, LAT--lateral articulation, MD--middle articulation, ASP--aspirated, BG--beginning voiced, EN--end voiced, FLP--flap articulation, LBL--labialized, PHRG-pharyngealized, RTRF--retroflex, TNCN--tongue constriction, VLRD--velarized, TRLL--trill, CLS--close, NEU--neutral, OP--open, FR--front, CT--center (MD--mid), BK--back, U--u-like off-glide, I--i-like off-glide, @--schwa-like off-glide.

1.25 Assessment of the value of phonological coding in computer-based generation of programmed learning.

The preceding sections have made it clear that the problem of the coding of phonological information has not been satisfactorily solved. The example given in 1.24 is unsatisfactory in many respects and it is unlikely that significantly more satisfactory solutions can be found at the present time. However, even if a truly satisfactory coding system were available, it is questionable whether its use would be of great value in the computerized generation of programmed language instruction.

The reason for this lies in the relatively small amount of data processing involved. Thus, the number of allophones coded is less than 100 for each of the two languages discussed here. The comparison of two sets of such a magnitude for the purpose of ascertaining their common members is a relatively simple task for a human being and could easily be accomplished once the appropriate information was assembled. It would therefore appear that, at least in so far as the establishment of similarity or differences between the allophones of only two languages is concerned, the use of the computer would be superfluous. Once the information has been systematized to the point at which it is suitable for input, the relevant decisions could easily be made without the computer.

On the other hand, the computer would be useful in obtaining such data on large numbers of language pairs, e.g. English vs. German, English vs. French, German vs. French, English vs. Spanish, German vs. Spanish, French vs. Spanish. However, such results would be of practical value only if a major commitment to programmed instruction were made by a large number of speech communities.

To sum up, the encoding of phonological information is both hard to attain and of little practical value. The purposes of computerized instruction would probably be better served if decisions as to phonological similarities and differences were made by human beings. Allophones of the target language could then be coded with regard to the extent and nature of the difficulties they posed for speakers of the base language. Such coding would be of real value since it could be used in the comparison of a set with approximately one hundred members and a set (the allophones of the potential vocabulary) with many thousands of members.

1.3 Orthography.

1.31 Handwriting.

The results of research in this area showed marked similarities to those obtained in phonology. In particular, the work by Halle and Eden on the distinctive features of handwriting proved to have exactly the same defects as the distinctive-feature approach to phonology: it failed to provide unambiguous solutions and tended to distort the important role of "non-distinctive" variants.

A second similarity consists in the fact that it was impossible to obtain a truly satisfactory alternative to the distinctive-feature approach. However, in the case of handwriting, this was due primarily to the fact that no thorough study of either the English or the Ukrainian system was available. Aside from the models presented to beginning students, little objective information could be obtained on the actual handwriting practices of the two communities. Since the acquisition of such information by independent research was beyond the scope of the present project, it was necessary to limit our activities to the consideration of the standard models--a step which severely limits the practical value of our results, since it is clear to even the most casual observer that the actual practice of many if not most of those who write English or Ukrainian shows significant deviations from the prescribed norms.

1.3.. A tentative coding of information on handwriting.

Appendix 2 shows the results of the application of our coding to the small handwritten letters of English and Ukrainian. Once again the format is that of an IBM card: a label (first character of *E*nglish or *U*krainian followed by the actual designation of the letter: a,b,c, a,b,v, etc.) occupies columns one through ten. The coding begins in column eleven.

The coding is based on the projection of the model for each letter on graph paper with 1/16" squares. The "absolute zero" of the vertical scale is the "bottom line" (i.e. line marking the "bottom" of *a*, *b*, *c*, etc.) of the orthographic model. No absolute zero is needed on the horizontal scale since all such specifications are adequately defined by reference to the "relative zero" defined by the initial point of the letter.

The coding for each letter begins with a specification of the vertical displacement of the initial point from absolute zero. This consists of the letter *v* (representing "vertical") and a following number (positive for points "above the line" negative for those below) representing the number of graph squares in the displacement. Thereafter, the coding represents combined vertical and horizontal (preceded by *h*) displacements from the point last defined. A new set of displacements is given each time the handwritten line reverses direction in terms of either the horizontal or the vertical axis of the graph. A positive vertical displacement indicates that the new point of reversal is above the preceding one, while a positive horizontal displacement shows that it is to the right of the latter. Negative displacements indicate a location below or to the left of the preceding point of reversal, respectively.

Unless interrupted by a period or a colon, the displacements determine a French curve such that the path of the handwritten line to and from a given point of reversal is determined both by the preceding and by following points of reversal. The use of a period after a displacement set signifies that the path of the handwritten line leading away from the reversal point specified by that set is independent of the preceding point of reversal (i.e. there is a "break" in the curvature). The colon (cf. English *i* and *t*) is used for the same purpose, and in addition indicates that no handwritten line joins the points of reversal specified by the preceding and following displacement sets. The closing parenthesis (cf. *i*, *j*) is used to indicate that the preceding displacement set does not specify a point of reversal in the handwritten line itself, but rather provides the information needed to determine the French curve leading *away from* the following point of reversal. It is used to allow the encoding of cases in which a break in the curvature occurs before three successive specification sets have been given. The letter *e* is used to indicate displacements one-half a graph square in magnitude.

1.312 Value of coded information on handwriting in computerized generation of programmed instruction.

The same considerations which led us to reject, at least in the case of work with only two languages, the encoding of phonological information apply to treatment of handwriting. On the one hand, our coding is arbitrary and misleading, since it cannot account for the multitude of variations which actually occur. On the other, it is unneeded, since a human being can easily decide which members are common to the two sets of handwritten letters and which are found in only one. While coded information on the results of such human information would be useful, it is wasteful to insist that the decisions themselves be made by the computer.

1.32 Printed letters.

The work on the development of coding for printed symbols showed parallels to that on handwriting. Similar solutions with similar defects were obtained. However, since the coding of information of printed symbols is of no more value in the computerized generation of programmed learning than the coding of information on handwriting, no examples will be given.

1.4 Morphology.

The work done in this area concentrated on the development of an unambiguous coding for flectional changes. This coding rested on two fundamental assumptions: 1) every inflected form of a given word can be viewed as a string of discrete graphic units, and 2) any such string of units can be obtained by one or more of a small number of processes (e.g., addition, deletion, insertion, omission, or modification) from another string of such units which represents the "basic form" of the given word.

1.41 The selection of graphic units.

It is clear that the graphic units used in the specification of basic forms and in the derivation of inflected forms must be in some way relatable to both the allophones occurring in spoken speech and to the written or printed letters of the standard orthography. In so far as such relations are complex the problem of teaching the language becomes more complicated. It was therefore decided to use graphic units bearing a one-to-one relationship to the letters of the standard orthography in dealing with Ukrainian forms. The forms of standard English were not dealt with.

1.42 The selection of basic forms.

For those Ukrainian categories (nouns, adjectives, verbs) which show a significant number of inflected forms the following were selected as basic forms: nouns--nominative singular, adjectives--nominative singular masculine, verbs--third person singular of the present tense. These selections were made in order 1) to simplify the relationship between the learning of basic forms and the learning of standard vocabulary lists; and 2) to insure that the basic form could participate in a large number of binary syntactic pairs.

1.43 Supplementary coding of basic forms.

In addition to the string of graphic units representing a basic form it was often necessary to give additional coding. Thus, every basic form is accompanied by a numeral indicating which vowel of the form bears the stress. Furthermore, cases in which neither the graphic units nor the stress num~~b~~eral can serve as reliable criteria for the generation of other inflected forms necessitate the inclusion of additional coding indicating which of a number of possible generative rules applies to the basic form in question.

Extensive examples of basic forms and the additional coding which accompanies them are found in the list of Ukrainian verbal types given in Appendix 3. Note that the stress of the basic form is indicated by the numberal in column *A*. This numeral indicates the vowel, counted from the end of the form, on which the stress falls (i.e. *1* indicates final stress, *2* penult stress, etc.). The next three columns (*IM*, *PS*, and *IN*) give information relative to addition and deletion in the imperative, past tense and infinitive, respectively, while the columns labelled *STR1* and *STR2* cover stress shifts. Finally, the columns labelled *O*, *C*, and *I* contain coding relevant to omissions, changes, and insertions, respectively.

1.44 The coding of rules for the derivation of other inflected forms.

In order to permit the comparison of the inflectional morphology of two different languages, strict procedures have been formulated concerning the statement of the relationship of a given inflectional form to the basic form. The possibilities are as follows:

Simple deletion (symbol *D*)--the final graphic unit(s) of the basic form are dropped (e.g. *lize*: *liz*).

Simple addition (symbol *A*)--one or more graphic units are added after the final graphic unit of basic form (e.g. *lize*: *lizesh*).

Combined deletion and addition--this consists of simple deletion followed by the addition of one or more graphic units to the results of the deletion.(e.g. *lize*: *liz*: *lizla*).

Stress shift forward (symbol *S1*)--the stress of the derived inflectional forms is one syllable to the right of that in the basic form (e.g. *hovoryt'*, *2*: *hovorju*, *1*).

Stress shift backward (symbol *S-1*)--the stress of the derived inflectional form is one syllable to the left of that of the basic form (e.g. *stryzhe*, *1*: *stryhla*, *2*).

(Note that in the case of a difference of more than one syllable between the position of the stress in the basic form and that in the derived inflectional form (e.g. *obiv*je*, *2*: *obvyla*, *1*) the symbols *S1* or *S-1* are used repeatedly.)

Insertion (symbol *I*)--two contiguous graphic units of the basic form are separated in the derived inflectional form by one or more inserted graphic units (*e.g.* *are*: *mer*).

Omission (symbol *O*)--two graphic units which, in the basic form are separated by one or more other graphic units become continguous in the derived inflectional form as the result of the omission of the intervening unit(s) (e.g. *bere*: *braty*).

Change (symbol *C*)--one or more graphic units within the basic form is/are replaced by the same number of (differing) graphic units in the derived inflectional form (e.g. *nese*: *nis*). Decisions as to whether a given difference between two basic forms should be coded as, on the one hand, combined deletion and addition, or, on the other, insertion, omission, or change (or a combination thereof) are to be based on the following rule: deletion and addition may not be used when this would entail the deletion and addition in one and the same position within a morpheme of one and the same graphic unit; they *must* be used in all other cases. Thus, we cannot view *bere*: *braty* as the deletion of *ere* followed by the addition of *raty*, since the *r* deleted is in exactly the same position (final) in the root as the *r* added. Here the analysis must be 1) deletion of *e* 2) addition of *aty* and 3) omission of *e* between *b* and *r*. For the same reason, the relationship between *nese* and *nis* must be analyzed as 1) deletion of final *e* and 2) change of the *e* between *n* and *s* to *i*. On the other hand, we must analyze the relationship of *vede* and *viv* as 1) the deletion of *ede* and 2) the addition of *iv*, since in this case the graphic units added do not include any of those deleted.

The coding strings resulting from the application to basic forms and their additional coding of a set of rules describing Ukrainian verbal flection are generated by the program documented in section 4.3 below and will discussed in connection with that program.

1.45 Value of coded information on inflectional morphology.

In so far as the comparison of the morphology of one language with that of another is concerned, there is little point in morphological coding. As in the case of phonology and orthography, the decisions as to similarities and differences in flectional patterns can be easily made by a human being who possesses the data necessary to formulate the coding on which the computer will operate. However, the difficulties in reaching such decisions with the help of the computer are much greater than in the two areas previously discussed, since the comparison of the inflection of two different languages entails not only the comparison of coding strings of the type discussed in section 4.3 below, but also a comparison of the relationship of the graphic units used in the descriptions of the inflection of both languages to the phonological and orthographic systems thereof.

On the other hand, inflectional coding does possess a real value in the computerized generation of programmed learning, in that it permits the comparison of the flectional patterns of different basic forms of the same language. Such comparisons can be used in two ways: 1) to establish whether the introduction of a given item at a given point entails the learning of a new flectional pattern; or 2) to determine whether the flectional pattern of one item is more or less complicated than that of other items. The use of inflectional comparisons for the first of these purposes is illustrated by the program documented in section 4.3 below.

No program has been written to illustrate the second use of such comparisons.

1.5 Syntax.

Research in this area was marked by the same frustrations which were spoken of in the preceding sections. The generative-grammar approach to syntax, like the distinctive-feature approach to phonology, offers no sound basis for the encoding of information to be used in a comparison of two languages. Recourse was therefore taken to more traditional approaches based on such concepts as that of the parts of speech (e.g. noun, verb, adjective) and syntactic pairs (subject-predicate, modifier-head. Unfortunately limitations of time and money made it necessary that work in this area be discontinued before useable results could be obtained.

1.6 Summary of research on coding.

Two important points should be emphasized in speaking of research in this area. The first of these is the fact that in every case "structural linguistics" failed to provide a suitable basis for the creation of objective coding systems. In particular, the distinctive-feature approach to phonology and handwriting, and the generative-grammar approach to syntax, with their emphasis on the interrelationships between the parts of a single system do not provide a reliable basis for the objective comparison of two or more systems. In view of the well-known inadequacies of earlier approaches, it is clear that the science of linguistics will have to make considerable progress before such comparisons will be feasible.

The second point is to a certain extent a consolation for the first: in none of the areas studied did the use of coded information to allow the computerized comparison of two languages prove of any great value in the generation of programmed learning. Such comparisons, in so far as they can be made at all, are easily carried out by human beings. It is only the results of these comparisons which, properly coded, can play a significant role in the computerization generation of programmed instruction.

2. Work on Lexicon.

The work on the basic lexicon was facilitated by the availability from other sources of a file of some 100,000 dictionary entries. This file already contains information on the inflectional patterns of individual items and on the parts of speech they represent. However, it proved impossible to complete the desired syntactic and lexical coding within the period allotted for the project. Work in this area was therefore abandoned in order to facilitate progress in the writing of the programs discussed under 4 below.

3. Program for the comparison of coded information on orthography and phonology.

3.1 General remarks.

In view of the fact that there is, as was pointed out in section 1, little point in having the computer reach decisions concerning similarities or differences in the structure of different languages, most of our programming effort was devoted to programs designed to permit the computer to order material about what such decisions had been made. However, a program (Appendix no. 4) has been written which operates on input of the type illustrated in Appendices 1 and 2 to establish which Ukrainian allophones and graphemes have exact English correspondents and which do not. Because of its relative simplicity this program will serve as a useful illustration of the programming language in which all the programs herein discussed have been written.

3.2 The SNOBOL programming language.

3.21 General comments.

The programming language employed is called SNOBOL III, a language designed especially for string manipulation. The original version of this language is discussed in an article by D.J. Farber, R.E. Griswold, and I.P. Polonsky in the Journal of the Association for Computing Machinery (Volume II, 1964, pp. 21-30). Supplementary information on the versions known as SNOBOL II and SNOBOL III may be found in Memorandum MAC-M-231 (April 9, 1965) published by Project MAC at the Massachusetts Institute of Technology. However, it is hoped that the following detailed description of the program in Appendix 4 will suffice to enable the reader to follow the documentation of the other programs included in this report.

3.22 Detailed description of program in Appendix 4.

3.221 Line format.

Each line of the program is divided into four sections. The left-most of these is reserved for labels (e.g. BEGIN, RDUKR) which have the same function as in other programming languages (i.e. serve as a symbol for the location of the line they accompany). The next column is reserved for string names, e.g. TARG, SAME, **or** functions, e.g. READ, PRINT, EQUALS. The third section includes 1) information or inquiries about the contents of **a single string** (e.g. =CLB "," BLB) **or 2)** string names which are the arguments of a given function (e.g. [TBY, BBY]). The final section, always preceded by a slash, contains information on transfers to program locations other than the immediately following line. The label of the location is enclosed in parentheses. A transfer to be made only if a test is successful has an S before the parentheses. A transfer to be made only upon failure of a test has an F in the same position, e.g. /F(GRENG). Unconditional transfers have no letter between the slash and the label, e.g. /(RDUKR).

The first line of the program starts with the label BEGIN. The second column contains the function READ which has the arguments UKR and 3. The effect of this line is to cause one record to be read off the tape mounted at station 3 every time the sequence UKR is mentioned in the second section of a line. The records on this tape will contain data on either Ukrainian allophones or Ukrainian graphemes in the card-image format shown in Appendices 1 and 2, respectively.

The next line has no label but begins with the function PRINT, followed by the arguments DIFF and 1. This line causes a tape record to be written on the tape mounted at station 1 every time the sequence DIFF is mentioned in the second column of a line. The contents of the record will be the contents of any string or strings which occur in the third column immediately after an equals sign (cf. the line labelled PRDIFF).

The second section of the line labelled RDUKR causes a record to be read from the tape at station 3 (cf. comments on BEGIN above). The third section (*TARG/"72"*) causes the first 72 characters of that record (i.e. the card image) to be stored in the computer as a string with the name TARG (i.e. data on the target language). The fourth section causes a transfer to the finish routine if no more records can be read from tape 3 (i.e. if the Ukrainian data is exhausted).

The second section of the following (unlabelled) line uses the string TARG (which itself remains unchanged) to generate 3 new strings: TLB (label of target data), containing the first five characters of the label section of the card image, TBLK (blank area in target data), containing the next five characters (always blanks!) of that section, and TBY (body of target data) containing the actual coded information on the target-language allophone or grapheme (i.e. the remaining 62 columns of the card image).

The line labelled RDENG is similar in function to BEGIN: it causes a record consisting of English data in card-image form to be read from the tape mounted at station 4 every time the sequence ENG is mentioned in the second section of a line.

The next two lines (GTENG and GTENG + 1) parallel RDUKR and RDUKR + 1.

The card image is stored as the string BASE and used to generate label, blank, and coding strings. Note that the fourth column of GTENG contains a transfer to the routine which causes the Ukrainian grapheme or allophone under consideration to be included in the list of those lacking English correspondents. This transfer takes place when no more English data can be read from tape 4 (i.e. when the Ukrainian allophone or grapheme has been compared with all of the English allophones or graphemes, respectively).

The function EQUALS in the second section of the next line compares the coding strings of the Ukrainian (TBY) and English (BBY) allophones or graphemes. If these are not identical a transfer is made to GTENG where another English card is read for comparison with the Ukrainian card image. If the coding strings are equal, the second section of the following line causes the generation of a **new string named** SAME, the contents of which are equal to the label of the Ukrainian grapheme or allophone (TAB) followed by a comma and the label of the English grapheme or allophone (e.g. UP1,EP2 or EA,UA). The next line causes the contents of SAME to be written as **a single tape** record on the system output tape (e.e. the tape mounted at station 6). Tape 6 thus contains the labels of Ukrainian allophones or graphemes accompanied by the labels of their English correspondents.

After the record has been written on tape 6 the line labelled RWENG causes the rewinding of the tape containing the English data.

This is followed by a transfer to RDUKR where the next Ukrainian card image is read and processed for comparison with the entire set of English card images.

The line labelled PRDFF, which is reached from GTENG after an unsuccessful attempt to read another English card image (i.e. when all the English card images have been compared unsuccessfully with a given Ukrainian card image), causes the entire Ukrainian card image to be written as a record on tape 1 (cf. the comments on the line following BEGIN). This is followed by a transfer to RWENG to prepare the English data for comparison with the next Ukrainian card image.

The line labelled FINISH, which is reached only from RDUKR after an unsuccessful attempt to read a Ukrainian card image (i.e. after all the Ukrainian card images have been compared with the English card images), rewinds tape 3. The next three lines rewind tapes 4, 6, and 1 respectively.

The line labelled END indicates to the compiler that the terminal line of the program has been reached and that the initial line is labelled BEGIN.

3.222 Supplementary comments on third section of lines.

As has been pointed out, when the second section of a line is occupied by a function (e.g. READ, PRINT, EQUALS, REWIND), the third section contains the arguments of that function. The latter are always enclosed in parentheses and separated by a comma. In the case of READ AND PRINT these arguments are 1) a string name (e.g. UKR, ENG) and 2) the number of a tape-drive station (e.g. 3,4). The use of the function READ causes a record to be read from the given tape every time the string name occurs in the second section of a line, while the use of PRINT causes a record to be written on the tape. Note that each argument of the functions READ and PRINT is enclosed in quotation marks. Quotation marks are used in the third section of a line to indicate the presence of literals.

The functions of EQUALS are the two string names (e.g. TBY, BBY) the contents of which are being compared.

The function of rewind is the string name (e.g. UKR, ENG) associated with the tape station to be rewound.

Note that the string names SYSPIT and SYSPOT are automatically associated by , the compiler with tape stations 5 and 6 respectively. The latter is used for output, the former for input.

If the second section of a line is occupied by a string name, the third section contains either a test of the contents of the string associated with that name or a modification of the contents of that string or both. The modification of the contents of a string is indicated by the use of the equals sign. Thus, in the line SAME = TLB "," BLB the contents of the string SAME is made equal to the contents of the string TLB (i.e. the label of the Ukrainian card image) followed by a comma and the contents of the string BLB (the label of the English card image). Note that the string names, unlike the comma, are not given as literals. (The line SAME = "TLB,BLB" would make the contents of SAME equal to the string of letters enclosed in quotation marks.)

The contents of a string is tested simply by omitting the equals sign at the beginning of the third section. It may be tested for the presence of a literal (e.g. SAME "B" meaning "Does the string SAME include a letter B?"), for the presence of the contents of another string (e.g. SAME TLB meaning "Does the string SAME include the contents of the string TLB?"), or for a string variable. The latter possibility represents an extremely powerful programming device and will therefore be discussed in some detail.

The presence of a string variable in the third section of a line is indicated by enclosing a string name in asterisks (e.g. *TARG*). Note that while the contents of a string variable cannot be specified, the number of characters it contains may be specified by following the string name by a slash and either 1) a literal number or 2) the name of a string the contents of which is a literal number. Thus *TARG/"72"* indicates that the string variable TARG contains 72 characters. On the other hand, if NMR = "6", then *TARG/NMR* indicates that TARG contains 6 characters.

A fundamental feature of string variables is that whenever a test for such a variable is successful, the portion of the string tested which matches the variable is made into a new string with the name of the given string variable. The string originally tested remains unchanged. Thus, SAME *TARG/"72"* tests the string SAME for the presence of 72 characters. As soon as 72 characters are found they are made the contents of a new string with the name TARG. (Note that any contents previously assigned to TARG is lost, so that the line in question is in effect a redefinition of the contents of TARG if 1) this string name has been used previously ,and 2) the string SAME is at least 72 characters in length. Note, however, that the contents of SAME are not modified.

Let us now turn once again to two of the lines of the program previously discussed in the hope of clarifying certain points which may still be unclear. The first such line is **the one labelled** RDUKR, the double function of which may have proved particularly troublesome. The use of the string name in section two causes a record to be read from tape 3 (c.f. BEGIN). This record constitutes the contents of the string UKR. The use of the string variable *TARG/"72"* in the third section causes the computer to test the contents of UKR to determine whether it contains at least 72 characters. If it does (and, since we are dealing with card images, it will), the first 72 characters encountered will become the contents of a string labelled TARG. In the next line the contents of TARG is examined for the presence of three consecutive variables of which the first two are five characters and the last 62 characters in length. Each of these is then made the contents of a new string.

If we wish to modify that part of the contents of a string in which a given test is successful, this may be done in the same line which contains the test simply by placing an equals sign after the variable(s), literal(s), or string(s) for the presence of which the string in section two is to be examined. Thus, TARG *TLB/"5"* = "USHCH" would test the string TARG for the presence of five characters. If they were encountered it would make them the contents of a string TLB. However the presence of the equals sign followed by "USHCH" would result in the first five characters of TARG itself being changed to USHCH. Such test-dependent modifications are particularly useful in deleting characters from a string--a process accomplished simply by leaving blank that portion of the third section of a line which follows the equals sign. Thus TARG *TLB/"5"* = results not only in a new string TLB, the contents of which are the first five characters of TARG, but also in the deletion of those first five characters from TARG itself.

Further uses of the string variables will be pointed out in the documentation of the other programs included in this report.

4. Programs designed to enable the computer to reach decisions regarding the desirability of introducing a given item at a given point.

4.1 Program to establish the orthographic difficulty of a given basic form.

4.11 Input.

The input consists of 1) a tape containing a list of basic forms (cf. the verbal basic forms given in Appendix 3) accompanied by their inflectional reinforcement and novelty values (cf. the program discussed in section 4.3), and 2) a list of Ukrainian letters which lack English correspondents and have not yet been used in at least two items previously introduced in the course. The second list (which might have been originally obtained as the output of the program discussed under 3 above) consists of the unfamiliar Ukrainian letters followed by a comma, a digit 0 or 1(indicating that the letter in question has not been introduced or has been used only once, respectively), and another comma (e.g. b,0,h,1,gh,0,je,0, etc). The last comma of the list is followed by a period (e.g. ja,0,.). Note that while each basic form constitutes a separate tape record, the entire list of unfamiliar letters is contained in a single record.

4.12 Operations carried out by the program (cf. Appendix 5).

The first three lines provide for the reading of the letters from tape 4, the reading of basic forms from tape 3 and the writing of output on tape 1, respectively. RDLS reads in the list of letters and generates therefrom a string L which contains all the letters and following commas, but lacks the final period. RDVB reads in a basic form and generates therefrom two strings: V, which contains the contents of the newly read tape record up to the first blank (enclosed in quotation marks), and E, which contains the contents of the record after that blank. A comparison with Appendix 3 shows that V contains the Ukrainian letters which constitute the basic form (e.g. BU[illegible]LYTJ) while E contains the additional coding.

The next line generates a new string VF, which is equal to V plus an immediately following period. The following two lines set the contents of the strings REI and NEW at zero. LSOUT forms a string LS which is equal to the contents of L plus an immediately following period.

The next line creates a string FL consisting of a single character found in VF. (Since tests are made from left to right this character will be the first character of VF.) At the same time, the presence of the equals sign followed by a blank causes the character which is the new contents of FL to be deleted from VF. The next line determines whether FL contains a period (i.e. whether all the characters in V have been processed). If the test is successful a transfer is made to ALDN. If not the program continues at TKLT which generates two new strings (TL and NM) from LS. TL contains the contents of LS up to the first comma (i.e. the first Ukrainian letter) while NM contains the contents of LS between the first and the second comma (i.e. the first digit). Note that the letter, the digit, and both commas are deleted from LS. If LS does not contain the items specified to the left of the equals sign (i.e. if the list of unfamiliar letters has been exhausted), the program transfers to LSOUT where the list of unfamiliar letters is reestablished and provided with a following period so that it may be compared with the next character of the basic form (i.e. the new contents of FL generated by the following line).

If the test for *TL* and *RM* is successful, the following line compares the contents of FL (i.e. the character from the basic form) with that of TL (i.e. the character from the list of unfamiliar letters). If they are not identical, the program returns to TKLT to obtain a new character from the list of unfamiliar letters. If they are identical, it proceeds to test the contents of RM for the presence of a 1 indicating that the given character has been previously used only once. A failure to find a 1 (a development indicating that the character has not yet been introduced) causes a transfer to CHKT. Success causes the contents of the reinforcement counter (REI) to be increased by one (line beginning with REI). This is followed by a transfer to LSOUT.

The line labelled CHKT increases the contents of the new character counter (NEW) by one and transfers to LSOUT.

The line labelled ALDN, which is reached only when all of the characters in the basic form have been compared with the list of unfamiliar letters (cf. comments on LSOUT + 2), generates a string NSTR the contents of which are the original contents of the tape record read from tape 3 (i.e. V " " E) followed by a period, the contents of the reinforcement counter, another period, and the contents of the new-character counter. The following line causes the contents of NSTR to be written as a record on tape 1 (cf. BEGIN + 2). This is followed by a transfer to RDVB where a new record is read from tape 3.

The program reaches the string labelled END only when no more records can be read from tape 3 (i.e. when all the basic forms have been processed).

4.13 Output.

The output of this program consists of a tape containing, in addition to the basic forms and supplementary coding mentioned in 4.11 above, 1) a period following by a number indicating how many characters of the basic form are unfamiliar characters which have been previously used only once, and 2) a second period followed by a number indicating how many characters of the basic form are unfamiliar characters which have not yet been introduced. Since characters belonging to the first group have a high value for purposes of reinforcement, the first number might be called the orthographic reinforcement value of the given basic form. Likewise, the second number might be referred to as the orthographic novelty value. It would not be unreasonable to assume that a basic form with a relatively high orthographic reinforcement value and a relatively low orthographic novelty value (other than zero) would be a more suitable choice for introduction into the course at a given point than a basic form with a relatively low orthographic reinforcement value and a relatively high orthographic novelty value.

4.2 Program to establish the probable phonetic and phonetic-orthographic difficulty of a basic form.

4.21 Input.

The input consists of 1) a tape containing a list of basic forms accompanied by their inflectional and orthographic reinforcement and inflectional values (cf. the programs discussed in sections 4.3 and 4.1, respectively); 2) a list of unfamiliar Ukrainian sounds which have not been previously encountered in more than one item; and 3) a list of unfamiliar Ukrainian letter-sound combinations (for the manner in which this list may be compiled see section 4.5) which have not been previously encountered in more than one item. The second list has the same format as the list of unfamiliar letters discussed in 4.11 (e.g.p2,0,b2,1,f2,0, etc.), and is likewise stored in the form of a single tape record. The third list consists of a series of Ukrainian letters each of which is immediately followed by an equals sign, the label (minus the identifying U) of a Ukrainian allophone, a digit (0 or 1 as in 4.11) and a second comma (e.g. p=pl,0,p=bl,0,b=bl,1, etc.). This list is also stored as a single tape record.

Note that, in contradistinction to the list of letters, neither the list or sounds nor the list of letter-sound combinations has a period as its last character.

4.22 Operations carried out by the program (cf. Appendix 6).

The first four lines provide for the reading of data from the three input tapes and the writing of output on tape 1. The next eight lines define the contents of strings used in various test routines in later parts of the program (cf. p. 14 ff). RDVB reads a record from the list of basic forms and makes it the contents of a string V. The next two lines insure that the previous contents of SW and ZW is blanked out. Next, the part of the newly read tape record which contains the Ukrainian basic form is made into the contents of a string LTPT. Then the contents of LTPT and a following period are made the contents of the string LTSC.

At GETWN the first character of LTSC is made the contents of a string WN. Next a test is made to see whether the contents of WN is a period (i.e. whether all the characters in LTPT have processed). If this is not so, WN is then compared with each of the letters of the Ukrainian alphabet until either a match is made or (after the comparison with JA on page 2) it is clear that the character in question is either JI or an apostrophe--neither of which has an independent sound value. In the latter case the current contents of WN is simply deleted from LTSC and the program transfers to GETWN where the next character of LTSC is made the contents of WN. If a match is made, the program transfers to one of the locations AA through JA. We shall illustrate the operation of the program at these locations by examining AA and JA in some detail. At AA (p.3) the contents of ACGOBK is defined as the label AA1 while that of UNAGBK is defined as AA2. The program then transfers to GE[illegible]STR (upper part of p.3)--a subroutine designed to determine whether a given vowel is stressed or unstressed. This is done by determining the number of vowels between the vowel in question and the end of the basic form. If this number plus one is equal to the number of the stressed vowel given in the coding accompanying the basic form, the vowel is stressed. Otherwise it is unstressed. Note that in the latter case the program transfers to ($UNAGBK). Here the dollar sign is used to indicate indirectness, i.e. the transfer is made to the location which is named by the contents of UNAGBK, in this case, AA2. Similarly if the vowel is stressed the program returns to the contents of ACGOBK (i.e. AA1).

At AA1 the contents of XW is increased by the addition of the (Ukrainian) allophone label A1 and a following comma. At AA2 the allophone added is A2. In either case a transfer is made to DLWN (bottom of page 2) where the contents of WN is deleted from LTSC. Next a plus sign is placed after the contents of XW. The last allophone label before the plus sign (in this case either A1 or A2) is then made the contents of a string SD. The contents of ZW is then increased by the contents of WN followed by an equals sign, the contents of SD and a comma (e.g. A=A1,). Next the plus sign is deleted from XW and the program transfers to GETWN.

At JA (p.3) the contents of DVTER is made equal to the label DLWN. AGGOBK and UNAGBK are then used with GETSTR to direct the program to JA1 or JA2, respectively. At JA1 the contents of six strings are defined so as to enable the proper returns from various subroutines, after which the program transfers to GETPRC (p. 13). Here the contents of LTSC except for the period is made the contents of PRDM, while the contents of LTPT preceded by a plus sign is made the contents of TSPR. Next that part of TSPR which precedes PRDM is made the content of RLPR. Finally, the contents of RLPR is increased by the addition of a following plus sign and the program transfers to the contents of PRGOBK which was defined at JA1 + 3 as TSPRCN (p. 14).

At TSPRCN (i.e. test for preceding consonant) the content of CNST is made equal to that of CNS (previously defined on page 1). At BRCN the first character of CNST is made the contents of TCNS and simultaneously deleted from CNST. (If no characters remain in CNST, the program transfers to the contents of NPRCN--defined at JA1 as JA11). At BRCN + 1 the last character of RLPR is compared with the contents of TCNS. If they match, the program transfers to the contents of ISPRCN--defined at JA1 + 4 as GETFLW (p. 14). If not, the program returus to BRCN for the next character from CNST.

GETFLW generates a string RLFL which contains that part of LTPT which follows the current contents of WR. The program then transfers to the contents of FLGOBK--defined at JA1 + 5 as TSTJ2 (p. 14). The latter is the beginning of a routine which determines whether or not the syllable following contains a sharp consonant. If so, the program transfers to the contents of FLSC, defined at JA1 + 2 as JA13. If no sharp consonant follows, the program transfers to the contents of FLHC--defined at JA1 + 1 as JA12.

When the program finally arrives at a location such as JA12 or JA13 (p.4), the proper allophone is added to the contents of XW and a transfer is made to the contents of DVTER--defined at JA as DLWN. Here the proper letter-sound combinations are added to ZW and preparations are made for the processing of the next character in LTSC.

Routines similar to those described in connection with locations AA and JA occupy the first sixteen pages of the program. By the time the period at the end of LTSC becomes the contents of WN (cf. GETWN + 1 on p.1) and the program transfers to TOTUP, XW contains all the allophones and ZW all the letter-sound combinations in the basic form under consideration.

At TOTUP (p. 17) a plus sign is added to XW. Next, the list of unfamiliar Ukrainian sounds is read from tape 3 (cf. BEGIN + 1) and made the contents of SNLST. At GTFSD the first Ukrainian allophone in XW is made the contents of FSD and deleted from XW. (If there are no more allophones in XW the program transfers to NOMOXW.) Next SNDLT is made equal to the contents of SNLST plus a period. At GTTSD the first unfamiliar character in SNDLT is made the contents of TSD, while the digit following it becomes the contents of SNM. Both items with their accompanying commas are deleted from SNDLT. (If SNDLT contains no more Ukrainian characters, the program transfers to GTFSD where the next character of **ZW** is made the contents of FSD.)

At GTTSD + 1 the contents of FSD and TSD are compared. (A failure returns the program to GTTSD for the next character in SNDLT.) If the comparison succeeds, the contents of SNM is compared to the digit 1. If this succeeds, the program transfers to UFSREI. Otherwise the contents of the new-sound counter (NW) is increased by one and the program transfers to GTFSD. At UFSREI the contents of the sound-reinforcement counter is increased by one and the program transfers to GTFSD.

The 9 lines beginning with NOMOXW exactly parallel the nine beginning with TOTUP. In this case, however, the letter-sound combinations of the basic form are compared with the unfamiliar letter-sound combinations of Ukrainian (cf. the program documented in section 4.5), and the results recorded in the sound-letter-reinforcement (RI) and new-sound-letter (NE) counters.

When the last sound-letter combination in ZW has been processed, the program transfers to NOMOZW where the contents of the string NWV is made equal to the tape record read at RD 9 plus the contents of the four counters (each set of contents preceded by a period). At NOMOZW + 1 this string is written as a record on tape 1 (cf. BEGIN + 3). The next four lines blank out the contents of NW, RE, NE, and RI, after which the program transfers to RDVB where a new record is read. When no more records remain on tape 4, the program transfers to END.

4.23 Output

The output of this program is a tape containing a list of basic forms with ,their accompanying coding (cf. Appendix 3) followed by 1) their inflectional reinforcement and novelty values (cf. section 4.3), 2) their orthographic reinforcement and novelty values (cf. section 4.1) 3) their allophonic reinforcement and novelty values, and 4) their letter-sound reinforcement and novelty values. Each value is preceded by a period.

4.3 Program to establish the probable inflectional reinforcement and novelty values of a basic form.

4.31 Input.

The input consists of 1) a tape containing a list of basic forms and their accompanying coding (cf. the verbal basic forms given in Appendix 3), and 2) a tape containing a list of the inflectional coding strings (for the generation of these see section 4.32) for the basic forms previously introduced. Each inflectional coding string is followed by a period and a digit 1 or 2. The digit indicates whether basic forms with the given inflectional coding have previously been used only once or more than once, respectively (e.g. AU,DV+AUEJ,DV+AMO+DV+ALA,DV+ATY,DV+.1) Note that both the basic forms of the first list and the inflectional coding strings of the second list constitute individual tape records. Note that the basic form tape is mounted at station 5.

4.32 Operations carried out by the program (cf. Appendix 7).

At BEGIN the previous contents of NVB (the string which will eventually contain the inflectional coding of the basic form being evaluated) is blanked out. At READIN a basic-form record is read from the system input tape (station 5). The first 72 characters of the record are simultaneously made the contents of a string V. At READIN +1 a test is made for a reflexive verb ending in SJA. If one is found the program transfers to RFL, a set of routines not included in this Appendix, but structurally parallel to the routines given for the treatment of non-reflexive verbs.

If the verb is non-reflexive, the program proceeds to P1, a routine for the generation of the inflexional coding for the first person singular of the present tense. The first test determines whether the Ukrainian basic form ends in YTJ (cf. the forms on pages 1, 2, and 3 of Appendix 3). If it does not, the program transfers to EP1 (middle of the following page). The next five lines determine whether the final YTJ is preceded by a labial consonant, in which case the program transfers to LP1 (following page). If no labial precedes, the next three lines test for a preceding sonant (transfer on success to SP1 on next page), while the following four test for a preceding palatal (transfer on success to PP1 on the next page). The last three lines test for a preceding T, D, or S, with transfer on success to TP1 (next page), DP1 (same page) or SSP1 (same page), respectively. If no transfer has been made, the final YTJ must be preceded by Z. Therefore at SSP1 - 1 a string NPR1 is given the contents AZHU,DCVC--inflectional coding for the processes "add ZHU after deleting the final consonant-vowel-consonant sequence (i.e. ZYTJ)."

The program next transfers to P1S on the following page. This routine first determines whether there is a 1 in column 32 (headed STR1 in Appendix 3) of the additional coding accompanying the Ukrainian basic form. If not, the program transfers to XP1 (two lines below) where the contents of NPR1 is made the contents of NVB whereupon the program transfers to P3 (eight lines below). If a 1 is encountered, the program proceeds to P1S + 1, where the inflectional coding S1 (indicating a shift of stress forward) is added to the contents of NPR1 (which becomes AZHU,DCVC,S1). The program then proceeds to XP1 and thence to P3.

If a successful transfer is made between P1+1 and SSP1 - 1 (on the first page) the subsequent operations parallel those initiated at SSP1 - 1, i.e. the proper coding for deletion and addition is placed in NPR1 and a transfer is made to P1S to determine whether the coding for a stress shift must be added. Only at DP1 (first page, bottom) and TP1 (second page, top) is a further test necessary to determine whether the D is preceded by Z or the T by S, respectively. In case of success the program transfers to ZDP1 or STP1 (both on the second page), respectively.

At EP1, which is reached only if the basic form does not end in YTJ (cf. P1), the basic form is tested for a final E (e.g. the forms on pages

3, 4, 5, 6, 7, 8, 9 of Appendix 3) with a transfer on failure to JEP1 (five lines below). If a final E is encountered, two more tests are made--one for an immediately preceding OR, the other for an immediately preceding I, both with transfer on success to EP11 where the proper coding for deletion and addition ("add JU after deleting the final vowel") is placed in NFR1 and a transfer is made to P1S. If both tests fail, the program proceeds to EP11 - 1, where different coding for deletion and addition ("add U after deleting the final vowel") is placed in NFR1 and the program transfers to P1S.

At JEP1, which is reached only if the basic form does not end in either YTJ or E (cf. P1 and EP1), the same form is tested for final JE (cf. the forms on pages 9, 10, and 11 of Appendix 3) with a transfer on success to EP11. If there is no final JE, the program proceeds to JEP + 1 where a test is made for final JITJ with a transfer on success to XP1. If this test fails, the form must end in STJ and the program proceeds to JEP1 + 2 where the proper coding for deletion and addition is placed in NFR1 and a transfer is made to XP1.

In addition to the inflectional coding for the first person singular of the present tense, generated by lines P1 (first page) through JEP1 + 2 (second page), the program generates inflectional coding for the third person plural of the present tense (line P3 on the second page through line DAP3 on the third page); the first person plural of the present tense (line PL on the third page through line

EPLP on the fourth page); the singular of the imperative (line I on the fourth page through line DAI on the sixth page); the masculine of the past tense (line PM on the sixth page through line CM on the twelfth page); the feminine of the past tense (line PF on the twelfth page through line CF on the eighteenth page); and the infinitive (line NTV on the sixteenth page through line DAN on the twenty-fourth page). The inflectional coding for each successive form is set off from that of the preceding form by a plus sign (cf. SP3 on the third page, SPL and SI on the fourth page, etc.).

After the inflectional ending for the infinitive has been made part of NVB (cf. SN on the eighteenth page), the program transfers to FINAL (on the twenty-fifth page). Here the previous contents of FVB is blanked out, the contents of CHGR is defined as the location WRPL, and TVB is given the same contents as NVB. At RTST the first set of inflectional coding (that for the first person singular of the present tense) is made the contents of R2 and simultaneously deleted from TVB along with the plus mark that separates it from the next set of inflectional coding. (If no set of coding followed by a plus is found in TVB, the program transfers to NOR2.)

At LSTR the contents of R2 is increased by a plus sign precding and a comma following the previous contents. At LSTR + 1 a test is made to determine whether R2 has an A as its first coding character (i.e. whether it represents the inflexional process "addition").

On failure the program transfers to NOA. If an initial A is encountered DLAD - 1 converts the addition coding into the contents of R22., while DLAD adds R22 to the contents of FVB. At DLAD + 1, R22 is deleted from R2 and the program transfers to YESA where a test is made to determine whether the next coding section of R begins with a D (i.e. a deletion instruction). Similar tests are used at NOA, GETS, NOS, NOI, and NOO to insure that when the program reaches WRPR, the inflectional coding of the basic form will be arranged in FVB in exactly the order in which inflectional coding is entered in the list of inflectional codings for verbs already introduced.

At WRPR the string FVB is increased by a following period. At WRPR + 1 provision is made to read in the list of inflectional coding for previously introduced basic forms from tape 4, while at WRPR + 2 (top of last page of program) provision is made to write output on tape 1.

At RDVTP a record is read from tape 4. The inflectional coding itself is made into the contents of CMVB while the accompanying digit becomes the contents of NMR. (If all the records have been read the program transfers to NOVT.) At RDVTP + 1, CMPR is given the contents of CMVB followed by a plus period. At RDVTP + 2 the coding string of the new basic form is compared to that of the record read from tape 4. On failure the program transfers to RDVTP where a new record is read from tape 4.

If the coding strings are identical a test is made at EQTST to determine whether only one basic form with, the given inflectional coding has been previously introduced. On success the program transfers to ONWR where the previous contents of V (i.e. the basic form read in at the beginning of the program) are increased by the addition of a following inflectional novelty value equal to 0 and an inflectional reinforcement value equal to 1, both preceded by periods. If the digit in NMR is not 1, the program proceeds to EQTST + 1 where the inflectional novelty value and the inflectional reinforcement values of V are both made equal to zero.

At NOVT, which is reached only from NOVTP after the inflectional coding of the basic form has been unsuccessfully compared with the entire contents of tape 4, V is given an inflectional novelty value equal to one and an inflectional reinforcement value equal to zero.

Each of the three lines which assign such values causes the program to transfer or proceed to STVERB where the new contents of V is written on tape 1 and the program transfers to READIN in order to read a new basic form from tape 5. END is reached when all the records on tape 5 have been read.

4.33 Output.

The output of this program is a tape containing a list of basic forms with their accompanying coding (cf. Appendix 3). Each basic form is followed by a period, the inflectional novelty value of the form, a second period, and the inflectional reinforcement value of the form.

4.4 Program illustrating the possible use of letter-sound and inflectional reinforcement and novelty values in the selection of a basic form for inclusion as the next form to be introduced at a given point (cf. Appendix 8).

4.41 Input.

The input consists of a tape containing the output of the program discussed in section 4.2 (cf. 4.23).

4.42 Operations performed by the program.

At BEGIN provision is made for the reading of input from tape 3 while BEGIN + 1 provides for the writing of output on tape 4.

At RDVB1 a record is read from tape 3 and its contents is split into V, containing the basic form with its accompanying coding, and R, containing the various novelty and reinforcement values. If no record can be read, a transfer is made to USEIT1. The next two lines make the contents of CHR and SHR equal to the locations RDVB1 and AFFR, respectively.

At SPLIT each of the eight novelty and reinforcement values **is made** into the contents of a separate line (A through H). At GH the previous contents of H (the letter-sound novelty value) is increased by the addition of a preceding period and a following plus sign. At GH + 1 a test is made for a zero novelty value with a transfer on failure to NZ. At GH + 2 HR is given the contents Z and the program transfers to GG. NZ tests for a novelty value equal to one with transfer on failure to NO. NZ + 1 puts an O in HR and transfers to GG. NO tests for a novelty value equal to two and transfers on failure to the contents of CHR--defined at RDVB1 + 1 as RDVB1--to read a new record from tape 3. NO + 1 puts a T in HR.

The lines GG through RW operate on the letter-sound reinforcement value in a fashion similar to the above, putting a Z, O, T or M into GR if the value is zero, one, two, or more than two, respectively. GT combines HR and GR into SR and transfers to the contents of SHR--defined in RDVB1 + 2 as AFFR. The latter and the three lines which follow it define the contents of FA, FB, FVR and FV, respectively. AFFR + 4 tests for a novelty rating of one and a reinforcement value of more than two. Success is followed by an immediate transfer to USEIT1.

Otherwise RDVB2 reads a record from tape 3 and assigns new contents to V and R. In the next two lines CHR and SHR are given the locations RDVB2 and AFSC, respectively, as their current contents, and the program transfers to SPLIT where the letter-sound novelty and reinforcement values are examined. (Note that a failure at NO now transfers the program to RDVB2 while the transfer from GT leads to AFSC).

At AFSC four lines generate the strings SA, SB, SVR, and SV which parallel FA, FB, FVR, and FV, respectively (cf. AFFR). Next SVR is tested for a novelty value equal to one and a reinforcement value equal to 3 or more, with a transfer on success to USEIT2. Otherwise the program proceeds to AFSC + 5, where a test is made to determine whether both of the basic forms being processed have identical letter-sound novelty and reinforcement values. If so, the program transfers to TSTVT; if not, the next line tests SVR for a novelty value equal to one. If this is found, a transfer is made to SVO where FVR is similarly tested. If FVR also has a novelty value equal to one a transfer is made to GTSCHF where a sequence of five lines tests the reinforcement values of SVR and FVR to determine which is greater. If that of FVR is greater, a transfer is made to RDVB2 where another basic form record is read for comparison with FV. If the reinforcement value of SVR is greater, a transfer is made to RPLFV, where four lines give to FA, FB, FVR, and FV the contents of SA, SB, SVR, and SV, respectively, after which a transfer is made to RDVB2.

If at SVO the novelty value of FVR is found to be other than one, the program transfers directly to RPLFV.

If at GTSCHF - 4 SVR has a novelty value other than one, the next line tests FVR for the same value, with a transfer on success to RDVB2. If this test also fails, GTSCHF - 2 tests SVR for a novelty value equal to zero, with a transfer on success to SVZ. Here FVR is tested for the same value, with a transfer on success to GTSCHF. If the test fails (i.e. the value is 2), SVZ + 1 and SVZ + 2 make the contents of ZGT and NZGT equal to the locations RPLFV and RDVB2, respectively.

At SVZGT three lines test for a reinforcement value of SVR equal to 3 or more, two, or one, respectively. If any test succeeds, a transfer is made to the contents of ZGT--defined at SVZ + 1 as RPLFV. Otherwise a transfer is made to the contents of NZGT--defined at SVZ + 2 as RDVB2.

If at GTSCHF - 2 the test fails (i.e. the novelty value of SVR is equal to two), the next line tests FVR for a novelty value equal to one, with a transfer on success to FVZ (page 3). Here the contents of ZGF is made equal to RDVB2 while that of NZGF is made equal to RPLFV--the opposite of what occurs at SVZ + 1. The program then transfers to SVZGT.

If at GTSCHF - 1 the test fails (i.e. both SVR and FVR have a novelty value equal to two), the program continues at GTSCHF.

At TSTVT, which is reached from AFSC + 5 only if SVR and FVR prove equal, the inflexional novelty values of the two basic stems are compared, with a transfer on success to TSTNS. Here a test is made for a letter-sound novelty value equal to zero, with a transfer on failure to FNOZ. At this point a test is made on FA for an inflexional novelty value of 1, with a transfer on success to RPLFV and on failure to RDVB2.

If at TSTNS the letter-sound novelty value is found equal to zero, the next line also tests FA for an inflexional novelty value of 1, but the transfer locations are reversed.

If at TSTVT the inflexional novelty values prove equal, the next line compares the inflexional reinforcement values, with a transfer on failure (possible only if the inflexional novelty value is zero) to TSTRS. Here FB is tested for a flexional reinforcement value equal to one, with a transfer on success to RDVB2 and on failure to RPLFV.

If the test at TSTVT + 1 is successful (i.e. if the inflexional and letter-sound reinforcement and novelty values of both basic forms are identical), the program proceeds to TSTNS and thence either to FNOZ or to TSTNS. From either of these points it then transfers to either RDVB2 or RPLFV, (a matter of complete indifference since both basic forms are equally desirable). At USEIT1, reached either from AFR + 4 on the discovery of a basic form with a letter-sound novelty equal to one and a letter-sound reinforcement value equal to three or more, or from RDVB1 or RDVB2 when no more records remain on tape 3, the contents of FV is written as a record on tape 4 (cf. BEGIN + 1). At USEIT2, reached only from AFSC + 4 after discovery of a basic form with the same values as those tested for at AFFR + 4, the contents of SV is written. END is reached from USEIT1 or USEIT2 after a record has been written.

4.43 Output.

The output consists of a single basic form which has been found to be either "ideally desirable" for introduction at the given point or, at least, as desirable as any of the other basic forms encountered. An "ideally desirable" basic form is defined as one with a letter-sound novelty value equal to one and a letter-sound reinforcement value equal to three or more. If no such basic form is encountered, the hierarchy of preference is as follows (N represents letter-sound novelty value, while R represents letter-sound reinforcement value): N=1, R=2; N=1, R=1; N=1, R=0; N=0, R=3+; N=0, R=2; N=0, R=1; N=2, R=3+; N=2, R=2; etc.

Two verbs with identical letter-sound novelty and reinforcement values are subjected to a test of their inflexional novelty and reinforcement values. In the case of a zero letter-sound novelty value, the verb with an inflexional novelty value equal to one is preferred to one with an inflexional novelty value equal to zero. Otherwise the verb with an inflexional novelty value equal to zero is preferred. If the verbs have identical inflexional novelty values, a verb with an inflexional reinforcement value equal to one is preferred to a verb with an inflexional reinforcement value equal to zero.

4.5 Input.

The input consists of four tapes. The first contains Ukrainian letter-sound combinations with the following format: UP=UP1,UP=UP2,UP=UB1, etc.

The second tape contains English letter-sound combinations in the same format. The third and fourth tapes contain output from the program discussed in section 3, i.e. a list of familiar Ukrainian allophones with their English counterparts and a list of familiar Ukrainian letters with their English counterparts, respectively. Note that each letter-sound combination or pair of allophones or letters constitutes a separate tape record.

4.52 Operations carried out by the program.

At BEGIN provision is made for the writing of the list of identical letter-sound combinations on tape 2. At RDUK a Ukrainian letter-sound combination is read from the system input tape (5). It is split into UL, containing the Ukrainian letter and US, containing the Ukrainian allophone. RDUK + 1 provides for the reading of English letter-sound combinations from tape 1. At GTENC such a combination is read and split into EL, containing the English letter and ES, containing the English sound. GTENC + 1 provides for the reading of familiar Ukrainian letters and their English correspondents from tape 3. At GTTSTL such a record is read and split into UTL, containing the label of the Ukrainian letter (e.g. UP), X, containing the blanks after the first blank in the Ukrainian label portion of the record, ETL, containing the label of the English letter (e.g. EN), and Y, also containing blanks.

GTTSTL + 1 compares UL and UTL, and transfers on failure to GTTSTL, where another record is read for comparison. A success is followed by a comparison of EL and ETL, with a transfer on failure to ENLT. GTTSTS - 1 provides for the reading of familiar Ukrainian allophones and their English correspondents from tape 4. At GTTSTS such a record is read and split into UTS, containing the label of the Ukrainian sound, and ETS, containing the label of the English sound. Next UTS is tested for the presence of one or more blanks, with a transfer on failure to NOUB. If a blank is found, the contents of UTS is reduced to include only that part of the label which proceeds the blank. At NOUB the same routine is applied to ETS.

At NOEB, US and UTS are compared, with transfer on failure to GTTSTS, where another familiar Ukrainian allophone is read for comparison. Success is followed by a comparison of ES and ETS with transfer on failure to RWSD.

If NOEB + 2 is reached, the Ukrainian and English letter-sound combinations must be identical, and the statement of this equivalence is made the contents of IDCN. At RWEC - 1 the contents of IDCN is written on tape 2 (cf. BEGIN) and the program transfers to RWEC where tapes 1, 3, and 4 are rewound, and a transfer is made to RDUK where the next Ukrainian letter-sound combination is read.

At RWSD, which is reached either from GTTSTS, when all of the familiar Ukrainian allophones have been compared to UL, or from NOEB + 1, when the English allophone ES, identical to US and UTS, is not also identical to ETS (and hence the letter-sound combinations must be different), tapes 3 and 4 are rewound and a transfer is made to GTECN where a new English letter-sound combination is read. At RWLT, which is reached either from GTTSTL, when all the familiar Ukrainian letters have been compared with UL, or from GTTSTL + 2, when the English letter EL, identical to UL and UTL, is not also identical to ETL, tape 3 is rewound and a transfer is made to GTECN.

At PRDFF, which is reached only from [illegible], when all of the English letter-sound combinations have been unsuccessfully compared to the Ukrainian combination, the Ukrainian combination followed by a comma is made the contents of NEW. At PRDFF + 1 NEW is written as a record on the system output tape (6).

END is reached from RDUK when no more Ukrainian letter-sound combinations remain to be processed.

4.53 Output.

The output consists of two tapes--one containing a list of familiar Ukrainian letter-sound combinations with their English equivalents, the other containing a list of unfamiliar Ukrainian letter-sound combinations. The latter tape, in slightly modified format, provides part of the input for the program discussed in 4.2 above.

4.6 Program to facilitate the evaluation of the desirability of a given basic form in terms of word derivation (cf. Appendix 10).

4.61 Input.

The input consists of two tapes, one containing a list of basic forms each of which is preceded by a six-digit number, the other containing a set of card images consisting of a six-digit number followed by a two-digit number and 16 pairs of two-digit numbers. The numbers preceding the basic forms on the first tape are in ascending numerical order. The six-digit number of each card image on the second tape is identical with one of the numbers preceding the basic forms of the first tape. Moreover, the first two-digit number of each of the sixteen pairs found on the card images of the second tape is identical with the last two digits of one of the numbers preceding the basic forms of the second tape. The second two-digit number of each of the pairs, as well as the two-digit number immediately following the six-digit number at the beginning of the card image, indicates the number of characters which precede the first character of the root in the basic form the six-digit number of which is indicated by either the first two-digit number of the pair or by the six-digit number respectively.

4.62 Operations carried out by the program.

At BEGIN provision is made to read the list of numbers from tape 1. At BEGIN + 1 the contents of AFRN is made equal to the location BRDF. At

RDN a card image of numbers is read and labelled N. Next N is split into F, containing the six-digit number and the two-digit number correlated therewith, and R, containing the sixteen pairs of two-digit numbers. The program then transfers to the contents of AFRN--defined at BEGIN + 1 as BRDF. Here the contents of AFRN is changed to TFBK. At RDF a basic form is read from the system input tape (5) and labelled A. Next A is split into B (the first four digits of the six-digit number), C (the last two digits of that number), D (a set of four blanks), E (the basic form itself) and G (the coding accompanying the basic form).

At TFBK F is tested for the letters "BLK", with a transfer on success to STTST. On **failure** (i.e. before the numbers in F have been processed), F is split into F1 (the six-digit number) and HF (the following two-digit number). Next the entire six-digit number preceding the basic form is placed in C. At STTST - 2 the contents of C is subtracted from F1 and the **result** placed in S. Next a test is made for a minus sign in S, in which case the program transfers to FOULUP.

If S is **positive** the program proceeds to STTST where F1 and C are compared, with a transfer on success to CHG. Since the first six-digit number on both input tapes is identical, the test will succeed in the case of the first basic form. At CHG F2 is given the contents of HF. At CHG + 1 F1 and HF are given new contents (i.e. the first and second two-digit numbers in R, respectively). After this the first four digits of R are deleted. If F1 and HF cannot be given new contents because there are no more digits in R, a transfer is made to RDN where a new card image is read.

Otherwise the program transfers to NSTST where F2 is tested for a double plus sign (the indication that the first character of the basic form coincides with the first character of the root), with a transfer on success to WRIT1.

On failure NSTST + 1 divides E into P (the number of characters specified by the contents of F2) and Q, (the remaining characters). NSTST + 2 makes NE equal to P and Q separated by a plus mark and is followed by a transfer to WTIT2.

Both WTIT2, which writes the newly segmented basic form on the system output tape (6), and WTIT1, which writes the unsegmented basic form preceded by a plus on the same tape, return the program to FGBK where F is given the contents "BLK". Next F1 is tested for a sequence of four characters, with a transfer on failure to RDF, where another basic form is read. If the four characters are found, FGBK + 2 deletes them from F1 (leaving only the last 2) and the program transfers to RDF.

The next time the program reaches TFBK the contents of F is equal to "BLK", so the lines dealing with the six-digit number are skipped and the program transfers to STTST where both F1 and C are two-digit numbers.

At RDN, which can be reached a second time only from CHG +1, when no more number pairs remain from the first card image, a new card image is read. This time however, the split into F and R is followed by a transfer to TFBK (defined as the contents of AFRN at BRDF), where the processing of the new six-digit number is begun.

At FOULUP, which can be reached only from STTST - 1, when C is larger than F1, the warning "OUT OF PHASE" is written on tape 6 to indicate that the two input tapes are not properly coordinated.

END is reached either from FOULUP or from EOF, when no more records remain to be read from the system input tape.

4.63 Output.

The output is a tape containing basic forms in which the first character of the root is preceded by a plus mark. This output can then be used as input for a right justification routine and a columnar sort routine to obtain an alphabetized list of the items in which any given root or set of roots occurs. Such a listing would be of help in evaluating the place of a given basic form within the overall word-derivation patterns of a given language.

5. Conclusions.

Because of the difficulties encountered in the research discussed in sections 1 and 2 above, it was not possible to test the programs discussed in section 4 on a Ukrainian lexicon or even to debug the programs themselves. Furthermore, it is clear that several more programs would be needed to permit the actual generation of a program outline--above all a program to modify the contents of the lists of unfamiliar letters and allophones and of familiar inflectional patterns once a new basic form has been selected for introduction at a given point. However, despite the fact that our original goal could not be fulfilled, the project may well be of value to future investigators, since it does, we believe, clear up a number of questions a misunderstanding of which might cost others considerable time and effort.

The principal insight gained is an acute awareness of the inability of present-day descriptive linguistics to provide a truly objective basic for the encoding of data on two different languages. Until this situation is improved, the assessment of differences and similarities between languages will remain an art best left to human beings.

A second point that has been established is that, once the differences between two languages are established, it is indeed possible to write programs designed to evaluate the desirability of introducing a given item at a given point. However, such programs will tend to be quite complicated and, as shown by the material in Appendix 3, necessitate extensive work on the preparation of input. Moreover, as the program discussed in 4.3 shows, many of the programs apply only to one language. It is therefore extremely doubtful that the computerized generation of a course outline can be realistically regarded as more economical than the generation of the same outline by human beings. In all probability future investigators would do well to concentrate their efforts on developing computer techniques for the implementation of a pre-existing course outline.

Appendix 1

	11	12	13	14	15	16					21					26					31					36					41					46			
	C	V	C	D	G	S	C	N	V	T	S	F	C	A	S	L	L	M			A	B	E	F	L	P	R	T	V	T	C	N	O	F	C	B	U	I	@
			P			T	T		D	N	H	I	H	C	Y	N	A	D			S	G	N	L	B	H	T	N	L	R	L	E	P	F	T	K			
						R				S	P	T	K	C	L	G	T				P			P	L	R	R	C	R	L	S	U							
																										G	F	N	D	L									
Up^1	1			1	1					1																													
p^2	1			1	1					1	1																												
b^1	1			1	1				1													1	1																
b^2	1			1	1				1		1											1	1																
f^1	1			1	1	1	1			1																													
v^1	1			1	1		1		1													1	1																
v^2	1			1	1	1	1		1		1											1	1																
m^1	1			1	1		1	1	1													1	1																
m^2	1			1	1		1	1	1		1											1	1																
t^1	1			1						1																													
t^2	1			1						1	1																												
f^2	1			1	1	1	1			1	1																												
d^1	1			1					1													1	1																
d^2	1			1					1		1											1	1																
s^1	1			1		1	1			1																													
s^2	1			1		1	1			1	1																												
z^1	1			1		1	1		1													1	1																
x^2	1			1		1	1		1		1											1	1																

	11 C	12 V	13 C P	14 D	15 G	16 S T R	C T	N	V D	T N S	21 S H P	F L T	C H K	A C C	S Y L	26 L N G	L A T	M D	31 A S P	B G	E N	F L P	L B L	36 P H R G	R T R F	T N C N	V L R D	T R L L	41 C L S	N E U	D P	F R	C T	46 B K	U	I	@
n^1	1			1			1	1	1											1	1																
n^2	1			1			1	1	1		1									1	1																
sh^1	1		1			1	1			1																											
sh^2	1		1			1	1			1	1																										
zh^1	1		1			1	1		1											1	1																
zh^2	1		1			1	1		1		1									1	1																
ch^1	1		1			1				1																											
ch^2	1		1			1				1	1																										
j^1	1		1			1			1											1	1																
j^2	1		1			1			1		1									1	1																
c^1	1			1		1				1																											
c^2	1			1		1				1	1																										
dh^1	1			1		1			1											1	1																
dh^2	1			1		1			1		1									1	1																
k^1	1		1		1					1																											
k^2	1		1		1					1	1																										
g^1	1		1		1				1											1	1																

	11	12	13	14	15	16				21						26			31					36					41					46			
	C	V	C	D	G	S	C	N	V	T	S	F	C	A	S	L	L	M	A	B	E	F	L	P	R	T	V	T	C	N	O	F	M	B	U	I	@
			P			T	T		D	E	H	L	H	C	Y	N	A	D	S	G	N	L	B	H	T	N	L	R	L	E	P	R	D	K			
						R				S	P	T	K	C	L	G	T		F			P	L	R	R	C	R	L	S	U							
																								G	F	N	D	L									
g^2	1		1		1				1		1									1	1																
x^1	1		1		1		1			1																											
x^2	1		1		1		1			1	1																										
gh^1	1		1		1		1		1											1	1																
h^1							1		1											1	1																
h^2							1		1		1									1	1																
h^3							1																														
l^1	1	1		1			1		1								1			1	1																
l^2	1	1		1			1		1		1						1			1	1																
r^1	1	1		1					1									1		1	1							1									
r^2	1	1		1					1		1							1		1	1							1									
o^1		1			1		1		1			1		1	1					1	1		1								1			1			
o^2		1			1		1		1			1		1	1					1	1		1								1		1	1			
o^3		1			1		1		1			1		1	1					1	1		1								1		1				
o^4		1			1		1		1			1			1					1	1		1						1					1			
o^5		1			1		1		1			1			1					1	1		1								1			1			
o^6		1			1		1		1			1			1					1	1		1								1		1				
o^7		1			1		1		1			1			1					1	1		1								1	1	1				

	11	12	13	14	15	16					21					26			31					36					41					46			
	C	V	C	D	G	S	C	N	V	T	S	F	C	A	S	L	L	M	A	B	E	F	L	P	R	T	V	T	C	N	O	F	M	B	U	i	@
			P			T	T		D	N	H	L	H	C	Y	N	A	D	S	G	N	L	B	H	T	N	L	R	L	E	P	R	D	K			
						R				S	P	T	K	C	L	G	T		P			P	L	R	R	C	R	L	S	U							
																								G	F	N	D	L									
o^8		1			1		1		1			1			1					1	1		1						1				1				
o^9		1			1		1		1			1			1					1	1		1						1			1	1				
y^1		1		1			1		1					1	1					1	1										1	1	1				
y^2		1					1		1						1					1	1								1			1	1				
a^1		1	1				1		1					1	1					1	1										1			1			
a^2		1	1				1		1					1	1					1	1										1		1	1			
a^3		1	1				1		1					1	1					1	1										1		1				
a^4		1	1				1		1						1					1	1										1			1			
a^5		1	1				1		1						1					1	1										1		1	1			
a^6		1	1				1		1						1					1	1										1		1				
e^1		1					1		1					1	1					1	1										1	1	1				
e^2		1					1		1						1					1	1									1		1	1				
i^1		1		1			1		1	1				1	1					1	1								1			1					
i^2		1		1			1		1						1					1	1									1		1					
i^3		1		1			1		1											1	1								1			1					

	11	12	13	14	15	16					21					26			31					36					41					46			
	C	V	C P	D	G	S T R	C T	N	V D	T N S	S H P	F L T	C H K	A C C	S Y L	L N G	L A T	M D	A S P	B G	E N	F L P	L B L	P H R G	R T R F	T N C N	V L R D	T R L L	C L S	N E U	O P	F R	C T	B K	U	I	6
i^4		1		1			1		1											1	1									1		1					
u^1		1		1	1		1		1			1		1	1					1	1		1							1				1			
u^2		1		1	1		1		1			1		1	1					1	1		1						1					1			
u^3		1		1	1		1		1			1		1	1					1	1		1						1				1	1			
u^4		1		1	1		1		1			1		1	1					1	1		1						1				1				
u^5		1		1	1		1		1			1			1					1	1		1							1				1			
u^6		1		1	1		1		1			1			1					1	1		1							1			1				
u^7		1		1	1		1		1			1			1					1	1		1							1		1	1				
u^8		1		1	1		1		1			1								1	1		1							1				1			
Ek^1	1		1		1					1									1																		
k^2	1		1		1					1																											
g^1	1		1		1				1											1	1																
g^2	1		1		1				1												1																
g^3	1		1		1				1											1																	
ng^1	1		1		1		1	1	1										1	1																	
p^1	1			1	1					1									1																		
p^2	1			1	1					1																											

	11	12	13	14	15	16					21					26			31					36					41					46			
	C	V	C	D	G	S	C	N	V	T	S	F	C	A	S	L	L	M	A	B	E	F	L	P	R	T	V	T	C	N	O	F	C	B	U	I	@
			P			T	T		D	N	H	L	H	C	Y	N	A	D	S	G	N	L	B	H	T	N	L	R	L	E	P	R	T	K			
						R				S	P	T	K	C	L	G	T		P			P	L	R	R	C	R	L	S	U							
																								G	F	N	D	L									
b^1	1			1	1															1	1																
b^2	1			1	1																1																
b^3	1			1	1															1																	
f^1	1			1	1	1	1			1																											
v^1	1			1	1	1	1													1	1																
v^2	1			1	1	1	1														1																
v^3	1			1	1	1	1													1																	
m^1	1			1	1		1	1	1											1	1																
t^1	1			1						1									1																		
t^2	1			1						1																											
t^3	1	1		1					1											1	1	1															
d^1	1			1					1											1	1																
d^2	1			1					1												1																
d^3	1			1					1											1																	
θ^1	1			1			1			1																											
$ð^1$	1			1			1		1											1	1																

	11	12	13	14	15	16					21					26					31					36					41					46			
	C	V	C̸ P	D	G	S T R	C T	N	V D	T N S	S H P	F L T	C H K	A C C	S Y L	L N G	L A T	M D			A S P	B G	E N	F L P	L B L	P H R G	R T R F	T N C N	V L R D	T R L L	C L S	N E U	O P	F R	C T	B K	U	I	@
ʒ^2	1			1			1		1														1																
ʒ^3	1			1			1		1													1																	
s^1	1			1		1	1			1																													
z^1	1			1		1	1		1													1	1																
z^2	1			1		1	1		1														1																
z^3	1			1		1	1		1													1																	
n^1	1			1			1	1	1													1	1																
ch^1	1		1			1				1											1																		
ch^2	1		1			1				1																													
j^1	1		1			1			1													1	1																
j^2	1		1			1			1														1																
j^3	1		1			1			1													1																	
sh^1	1		1			1	1			1																													
zh^1	1		1			1	1		1													1	1																
zh^2	1		1			1	1		1														1																
zh^3	1		1			1	1		1													1																	
ii^1		1		1			1		1	1				1	1							1	1									1		1				1	

	11	12	13	14	15	16					21					26			31					36					41					46			
	C	V	C	D	G	S	C	N	V	T	S	F	C	A	S	L	L	M	A	B	E	F	L	P	R	T	V	T	C	N	O	F	C	B	U	I	@
			P			T	T		D	N	H	L	H	C	Y	N	A	D	S	G	N	L	B	H	T	N	L	R	L	E	P	R	T	K			
						R				S	P	T	K	C	L	G	T		P			P	L	R	R	C	R	L	S	U							
																								G	F	N	D	L									
ii^2		1		1			1		1	1				1	1	1				1	1									1		1				1	
i^1		1		1			1		1					1	1					1	1										1	1					
i^2		1		1			1		1					1	1					1	1										1	1					1
ee^1		1					1		1	1				1	1					1	1									1		1				1	
ee^2		1					1		1	1				1	1	1				1	1									1		1				1	
e^1		1					1		1					1	1					1	1										1	1					
e^2		1					1		1					1	1					1	1										1	1					1
aa^1		1	1				1		1					1	1					1	1									1		1					
aa^2		1	1				1		1					1	1	1				1	1									1		1					1
a^1		1	1		1		1		1					1	1					1	1										1		1				
a^2		1	1		1		1		1					1	1	1				1	1										1		1				1
qq^1		1			1		1		1					1	1					1	1					1				1			1				
qq^2		1			1		1		1					1	1	1				1	1					1				1			1				1
q^1		1			1		1		1					1	1					1	1										1		1	1			
q^2		1			1		1		1					1	1	1				1	1										1		1	1			
uu^1		1		1	1		1		1			1		1	1					1	1		1							1				1	1		
uu^2		1		1	1		1		1			1		1	1	1				1	1		1							1				1	1		

	11	12	13	14	15	16					21					26			31					36					41					46			
	C	V	C P	D	G	S T R	C T	N	V D	T N S	S H P	F L T	C H K	A C C	S Y L	L N G	L A T	M D	A S P	B G	E N	F L P	L B L	P H R G	R T R F	T N C N	V L R D	T R L L	C L S	N E U	O F	F R	C T	B K	U	I	@
u^1		1		1	1		1		1			1		1	1					1	1		1								1			1			
u^2		1		1	1		1		1			1		1	1					1	1		1								1			1			1
oo^1		1			1		1		1			1		1	1					1	1		1							1				1	1		
oo^2		1			1		1		1			1		1	1	1				1	1		1							1				1	1		
o^1		1			1		1		1					1	1					1	1										1			1			
o^2		1			1		1		1					1	1					1	1										1			1			1
ai^1		1	1		1		1		1					1	1					1	1										1			1		1	
ai^2		1	1		1		1		1					1	1	1				1	1										1		1			1	
au^1		1	1		1		1		1					1	1					1	1										1		1		1		
au^2		1	1		1		1		1					1	1	1				1	1										1		1		1		
w^1		1		1	1		1		1			1								1	1		1							1				1			
y^1		1		1			1		1											1	1									1		1					
uux^1		1		1			1		1			1			1					1	1		1							1				1	1		
ux^1		1		1			1		1			1			1					1	1		1								1			1			
oox^1		1					1		1			1			1					1	1		1							1				1	1		
ox^1		1					1		1						1					1	1										1			1			
aix^1		1	1				1		1						1					1	1										1		1			1	

	11	12	13	14	15	16					21					26			31					36					41					46			
	C	V	C P	D	G	S T R	C T	N	V D	T N S	S H P	F L T	C H K	A C C	S Y L	L N G	L A T	M D	A S P	B G	E N	F L P	L B L	P H R G	R T R F	T N C N	V L R D	T R L Z	C L S	N E U	O P	F R	C T	B K	U	I	@
aux^1		1	1				1		1						1					1	1										1		1		1		
iix^1		1		1			1		1						1					1	1									1		1				1	
ii^1		1		1			1		1						1					1	1										1	1					
eex^1		1					1		1						1					1	1									1		1				1	
ex^1		1					1		1						1					1	1										1	1					
eax^1		1	1				1		1						1					1	1									1		1					
ax^1		1	1				1		1						1					1	1										1		1				
qqx^1		1					1		1						1					1	1					1				1			1				
qx^1		1					1		1						1					1	1										1		1				
1^1	1	1		1			1		1								1			1	1						1										
1^2	1	1		1			1		1								1			1	1																
1^3	1	1		1			1		1						1		1			1	1																
r^1	1	1		1			1		1									1		1	1					1											
r^2	1	1		1			1		1					1	1			1		1	1					1											
r^3	1	1		1			1		1						1			1		1	1					1											

appendix 2

ENGLISH	a	v4 vlh-1 v-4h-3 v-lhl v4h3. v-3h-1 v-lhe vlhle.
	b	v0 v9h6 vlh-1 v-9h-2 v-lhl v3hl. v-lhl vlh2.
	c	v4 vlh-1 v-4h-3 v-lhl vlh2.
	d	v4 vlh-1 v-4h-3 v-lhl v10h5. v-9h-3 v-lhe vlhle.
	e	v0 v4h3 vlh-1 v-3h-1 v-2hl vlh2.
	f	v0 v9h6 vlh-1 v-14h-5 v-lhl v2hl v2h-e. v-lhl vlh2.
	g	v4 vlh-1 v-4h-3 v-lhl v4h3. v9h3 vll·1 v5h4.
	h	v0 v9h6 vlh-1 v-10h-3. v5h4 v-lhe v-3h-le v-lhl vlh2.
	i	vl) v-lhl v5h3. v-4h-1 v-lhl vlh2: v5h-2.
	j	vl) v-lhl v5h3. v-10h-3 vlh-1 v5h5: v5h0.
	k	v0 v9h6 vlh-1 v-10h-3 v5h3. v-lhl v-2h-2 vlh-1 v-3h2 vlh2.
	l	v0 v9h6 vlh-1 v-9h-2 v-lhl vlh2.
	m	v0 v5h3 v-lhl v-4h-1. v5h3 v-lhl v-4h-1. v5h3 v-lhl v-3h-1 v-lhl vlh2.
	n	v0 v5h3 v-lhl v-4h-1. v5h3 v-lhl v-3h-1 v-lhl vlh2.
	o	v4 vlh-1 v-4h-3 v-lhl v4h3. v-lh2 vlhl.
	p	vl) v-lhl v5h3. v-10h-4. v10h7 v-lhl v-4h-3 v2h-1.
	q	v4 vlh-1 v-4h-3 v-lhl v4h3. v-8h-2 v-lhl v2hl v3h-e. v-lhl vlh2.
	r	vl) v-lhl v5h3. v-lhl vlh2. v-4h-1 v-lhl vlh2.
	s	vl) v-lhl v5h3. v-3hl v-2h-2 vlh-1. v-lh2 vlh2.
	t	vl) v-lhl v10h6. v-9h-3 v-lhl vlh3: v7h-4 v0h4.
	u	vl) v-lhl v5h3. v-4h-1 v-lhl v5h3. v-4h-1 v-lhl vlh2.

ENGLISH	v	v3 v2h2 v-1h1 v-3h-2 v-1h1 v5h4. v-1h1 v1h2.
	w	v1) v-1h1 v5h3. v-4h-1 v-1h1 v5h3. v-4h-1 v-1h1 v5h3. v-1h1 v1h2.
	x	v3 v2h2 v-1h1 v-3h-2 v-1h1 v1h2: v4h-1 v-5h-4.
	y	v3 v2h2 v-1h1 v-3h-2 v-1h1 v5h4. v-10h-4 v1h-1 v5h4.
	z	v3 v2h2 v-1h1 v-4h-2 v1h2 v-1h1 v-5h-3 v1h-1 v5h5.
UKRANIAN	a	v4 v1h-1 v-4h-3 v-1h1 v4h3. v-3h-1 v-1he v1h1e.
	b	v4 v1h-1 v-4h-3 v-1h1 v10h6 v-1h1 v1h1.
	v	v3 v2h2 v-1h1 v-3h-2 v-1h1 v5h4. v-1h1 v1h2.
	g	v3 v2h2 v-1h1 v-3h-2 v-1h1 v1h2.
	d	v4 v1h-1 v-4h-3 v-1h1 v4h3. v-9h3 v1h-1 v5h4.
	e	v0 v4h3 v1h-1 v-3h-1 v-2h1 v1h2.
	je	v4 v1h-1 v-4h-3 v-1h1 v1h2: v2h-2 v0h2.
	zh	v4 v1h1 v-2h1 v-4h-3. v1h-1) v-1h1 v5h4. v-5h-2. v5h4 v-1h1. v1h-1 v-3h-3 v-2h1 v1h1.
	z	v3 v2h2 v-1h1 v-4h-2 v1h2 v-1h1 v-5h-3 v1h-1 v5h5.
	y	v5 v-4h-1 v-1h1 v5h3. v-4h-1 v-1h1 v1h2.
	i	v1) v-1h1 v5h3. v-4h-1 v-1h1 v1h2: v5h-2.
	ji	v1) v-1h1 v5h3. v-4h-1 v-1h1 v1h2: v5h-2: v0h1.
	j2	v5 v-4h-1 v-1h1 v5h3. v-4h-1 v-1h1 v1h2: v5h-5 v-1h1 v1h2.
	k	v5 v-5h-3. v4h6 v1h-1 v-3h-3. v-eh1 v-1h-e v-eh1 v1h2.
	l	v1 v-eh-e v-ehe v5h3. v-4h-1 v-1h1 v1h2.
	m	v1 v-eh-e v-ehe v5h3. v-5h-1. v5h3. v-4h-1 v-1h1 v1h2.

UKRAINIAN	n	v5 v-5h-3. v2h2: v3h-3) v-3h3 v3h3. v-4h-1 v-1h1 v1h2.
	o	v4 v1h-1 v-4h-3 v-1h1 v4h3. v-1h2 v1h1.
	p	v0 v5h3 v-1h1 v-4h-1. v5h3 v-1h1 v-3h-1 v-1h1 v1h2.
	r	v5 v-10h-6. v10h7 v-1h1 v-3h-1 v-1h1 v1h2.
	s	v4 v1h-1 v-4h-3 v-1h1 v1h2.
	t	v0 v5h3 v-1h1 v-4h-1. v5h3 v-1h1 v-4h-1. v5h3 v-1h1 v-3h-1 v-1h1 v1h2.
	u	v3 v2h2 v-1h1 v-3h-2 v-1h1 v5h4. v-10h-4 v1h-1 v5h4.
	f	v4 v1h-1 v-4h-3 v-1h1 v10h4. v-15h-5. v10h5 v-2h1 v-3h-3 v1h-1 v2h1. v-2h-1 v-1h1 v1h2.
	x	v4 v1h1 v-3h1 v-2h-2 v1h-1: v4h6 v1h-1 v-3h-2 v-2h1 v1h1.
	c	v5 v-4h-1 v-1h1 v5h3. v-4h-1 v-1h1 v1h2. v-3h-1e v1h-e v3h3.
	ch	v4) v-1h1 v2h1. v-1h1 v1h2. v-4h-1 v-1h1 v1h2.
	sh	v5 v-4h-1 v-1h1 v5h3. v-4h-1 v-1h1 v5h3. v-4h-1 v-1h1 v1h2.
	shch	v5 v-4h-1 v-1h1 v5h3. v-4h-1 v-1h1 v5h3. v-4h-1 v-1h1 v1h2. v-3h-1e v1h-e v3h3.
	ju	v5 v-5h-3. v2h1. v3h-5) v-3h5 v3h5 v-1h1. v1h-1 v-4h-3 v-1h1 v4h3. v-1h2 v1h2.
	ja	v1 v-1h1 v4h2e v1h-1 v-2h-1 v-1h1 v2h1. v-4h-1 v-1h1 v1h2.
	j1	v5 v-4h-1 v-1h1 v1eh1 veh-e v-eh-1.

Appendix 3

	A	IM	FS	IN	STR 1	STR 2	O	C	I
BURLYTJ	1								
BORONYTJ	2				1				
BALAGURYTJ	2								
VYKORENYTJ	4	1							
BOLYTJ	1		1						
VIGORYTJ	3	1	1						
VISHYTJ	1								
VOLOCHYTJ	2				1				
ZDOVZHYTJ	2								
VYRISHYTJ	3	1							
TJAZHYTJ	1		1						
DERZHYTJ	2		2		1				
ZAMOVCHYTJ	2		2						
BRJAZHCHYTJ	1		2						
VYLEZHYTJ	3	1	2						
BIZHYTJ	1		3						
VYBIZHYTJ	3	1	3						
SUCHYTJ	2		4		1				
GROZYTJ	1								
ZARAZYTJ	2				1				
ZBLYZYTJ	2								
[illegible]	4	1							

	A	IM	PS	IN	STR1	STR 2	O	C	I
BASYTJ	1								
GASYTJ	2				1				
ZMUSYTJ	2								
VYGOLOSYTJ	4	1							
VIDYTJ	1		1						
VYSYTJ	2		1						
TVERDYTJ	1								
BLUDYTJ	2				1				
OBLAGORODYTJ	2								
VYNAGORODYTJ	5	1							
SMERDYTJ	1		1						
DOGLJADYTJ	2		1						
GNIZDYTJ	1								
JIZDYTJ	2								
BUTYTJ	1								
ZSAGATYTJ	2				1				
BALAMUTYTJ	2								
VYZOLOTYTJ	4	1							
BELJKOTYTJ	1		1						
OBVERTYTJ	2		1		1				
VYLETYTJ	3	1	1						
[illegible]	1								

	A	IM	PS	IN	STR1	STR2	O	C	I
MASTYTJ	2				1				
VYCHYSTYTJ	3								
VYROSTYTJ	3	1							
XRUSTYTJ	1		1						
VYSVISTYTJ	3	1	1						
BOMBYTJ	1								
DAVYTJ	2				1				
KVAPYTJ	2								
VYLOMYTJ	3	1							
SVERBYTJ	1		1						
PERETERPYTJ	2		1						
DOTERPYTJ	2		1		1				
VYTERPYTJ	3	1	1						
SPYTJ	1		2						
VYSPYTJ(SJA)	2	1	2						
BEREZHE	1		1						
STRYZHE	1				3				
DOPOMOZHE	2		1		1				
PEREMOZHE	2		1		2				
LJAZHE	2		1		4				
VYPRJAZHE	3	1	1						

	A	IM	PS	IN	STR1	STR2	O	C	I
VYSTRUZHE	3	1	2						
V*JAZHE	2		3		1				
VYV*JAZHE	3	1	3						
MAZHE	2		3						
BRESHE	2				1				
DYSHE	2								
VYBRESHE	3	1							
KOLYSHE	2		1		1				
VYKRESHE	3	1	1						
TOVCHE	1								
VYTOVCHE	3	1							
SICHE	1				3				
VYSICHE	3	1							
VOLOCHE	1		1						
VYVOLOCHE	4	1	1						
TCHE	1		2						
SKACHE	2		2		1				
KLYCHE	2		2						
VYTCHE	2	1	2						
KVOKCHE	2		3		1				
VYSMOKCHE	3	1	3						
XOCHE	2		4						

	A	IM	PS	IN	STR1	STR2	O	C	I
PLESHCHE	2				1				
VYPLESHCHE	3	1							
SVYSHCHE	2		1		1				
VYSVYSHCHE	3	1	1						
GREBNE	1								
TONE	2				1				
KANE	2								
VYSNE	2	1							
ZBLIDNE	2	1	3						
OBSJAGNE	1		2						
ODJAGNE	2		2		1				
ZMERZNE	2	1	2						
ZIV*JANE	2		4						
KLJANE	1		4	1					
STOGNE	2		1		1				
ZHNE	1		5						
POCHNE	1		5		4				
VYZHNE	2	1	5						
ZIZHNE	1		5				1		
OBIZHNE	2		5		1		1		
MNE	1		6						
VYMNE	2	1	6						

	A	IM	PS	IN	STR1	STR2	O	C	I
VIDIPNE	1		6		4				
PNE(TJSJA)	1		6	1	4				
VIDIPNE	1		6	1	4		1		
PERETNE	1		7		4				
VIDITNE	1		7		4		1		
ROZITNE	2		7		1	4	1		
ZITNE	1		7		4		2		
ZHENE	1		1				1		
VYZHENE	3	1	1				1		
VIDZHENE	1		1				1		1
ZHME	1								
VYZHME	2	1							
VIDIZHME	1						1		
OBIZHME	2				1		1		
JME	1		1		4				
VYJME	2	1	1						
DIJME	2		1		1	4			
VIDNIME	2		2		1				
DME	1		3		4				
VYDME	2	1	3						
VIDIDME	1		3		4		1		

	A	IM	PS	IN	STR1	STR2	O	C	I
DERE	1								
VYDERE	3	1							
MRE	1								
VYTRE	2	1							
VIDIMRE	1						1		
OBIPRE	2				1		1		
ZIPRE	1						2		
ORE	1		1						
VYORE	3	1	1						
BERE	1		1				1		
VYBERE	3	1	1				1		
VIDBERE	1		1				1		1
DIBERE	1		1				1	1	
UBERE	1		1				1		2
BORE	2		2		1				
VYBORE	3	1	2						
STELE	2				1		1		
VYSTELE	3	1					1		
SHLE	1								
VYSHLE	2	1							
NADISHLE	2				1				

	✓	A	IM	PS	IN	STR1	STR2	O	C	I
KAPLE		2		1						
SYPLE		2	2	1						
KOLE		2		2		1				
VYKOLE		3	1	2						
MELE		2		2		1				
VYMELE		3	1	2						
GUDE		1								
VIDPADE		1				3				
KRADE		2								
VYPADE		3	1							
BUDE		2			1	4				
BREDE		1		1						
VYBREDE		3	1	1						
SJADE		2		2						
ZHDE		1		3						
VYZHDE		2	1	3						
DIZHDE(TJSJA)		2		3		1				
JIDE		2		4						
IDE		1		5						
VYJDE		2	1	5						
VIDIJDE		2		5		1				
GUDE		1		6						
CVITE		1								

	A	IM	PS	IN	STR1	STR2	O	C	I
VYCVITE	3	1							
VIDROSTE	1							1	
VYROSTE	3	1						1	
METE	1		1						
VYMETE	3	1	1						
ZHYVE	1		3						
VYZHYVE	3	1	3						
PLYVE	1								
VYPLYVE	3	1							
REVE	1							1	
PLYVE	1		3	1					
VYPLYVE	3	1	3	1					
ZVE	1		1						
VYZVE	2	1	1						
OBIZVE	2		1		1				
OZOVE(TJSJA)	1		1				1		
REVE	1		2						
SKUBE	1				3				
LIZE	2								
B*JE	1								
VYB*JE	2								

	A	IM	PS	IN	STR1	STR2	O	C	I
VIDIB*JE	1						1	1	
VIDIB*JE	2				1		1		
V*JE	1				4				
ZIV*JE	1				4		1		
OBIV*JE	2				1	4	1		
LLJE	1				4				
VYLLJE	2								
VIDILLJE	1				4		1		
OBILLJE	2				1	4	1		
UVILLJE	1				4		2		
VYGRAJE	1								
ZNAJE	2								
GNYJE	1								
BRYJE	2								
GNYJE	1			1					
GNYJE	1			1	3				
CHERVONIJE	2								
OSMIJE(TJSJA)	1		1						
BAJE	2		1						
VZUJE	2		1						
KUJE	1								

	A	IM	PS	IN	STR1	STR2	O	C	I
GAMUJE	2								
VYKUJE	3								
VYSMIJE	3		1						
STOJITJ	1	1	1						
VYSTOJITJ	3	1	1						
GNOJITJ	1								
ZGNOJITJ	2				1				
UGNOJITJ	2								
DOPOJITJ	2	1			1				
UPOKOJITJ	2	1							
JISTJ									
VYJISTJ									
DASTJ					4				
VYDASTJ									
DOPOVISTJ					4				
DOPOVISTJ					4	1			

Appendix 4

```
BEGIN    READ     ("UKR","3")
         PRINT    ("DIFF","1")
RDUKR    UKR      *TARG/"72"*                                  /F(FINISH)
         TARG     *TLB/"5"* *TBLK/"5"* *TBY/"62"*
RDENG    READ     ("ENG","4")
GTENG    ENG      *BASE/"72"*                                  /F(PRDFF)
         BASE     *BLB/"5"* *BBLK/"5"* *BBY/"62"*
         EQUALS   (TBY,BBY)                                    /F(GTENG)
         SAME     = TLB "," BLB
         SYSPOT   = SAME
RWENG    REWIND   (ENG)                                        /(RDUKR)
PRDFF    DIFF     = TARG                                       /(RWENG)
FINISH   REWIND   (UKR)
         REWIND   (ENG)
         REWIND   (SYSPOT)
         REWIND   (DIFF)
END      BEGIN
```

Appendix 5

```
BEGIN     READ      ("DFLT","4")
          READ      ("VBFM","3"
          PRINT     ("NVBF","1"
RDLS      DFLT      *L* "."
RDVB      VBFM      *V* " " *E*                                  /F(END)
          VF        = V "."
          REI       = 0
          NEW       = 0
LSOUT     LS        = L "."
          VF        *FL/"1"* =
          FL        "."                                          /S(ALDN)
TKLT      LS        *TL* "," *NM* "," =                          /F(LSCUT)
          EQUALS    (FL,TL)                                      /F(TKLT)
          NM        "1"                                          /F(CHKT)
          REI       = REI + "1"                                  /(LSOUT)
CHKT      NEW       = NEW + "1"                                  /(LSCUT)
ALDN      NSTR      = V " " E "." REI "." NEW
          NVEF      = NSTR                                       /(RDVB)
END       BEGIN
```

Appendix 6

```
BEGIN   READ    ("VRB","4")

        READ    ("NSDS"."3")

        READ    ("NSTS","2")
        PRINT   ("RTG","1")
        VW      = "A,JA,E,JE,I,JI,Y,JU,U,O,"

        CNS     = "B,V,G,GH,D,ZH,Z,J2,K,L,M,N,P,R,S,T,F,X,C,CH,SH,SHCH,"

        SVW     = "JA,JE,JI,JO,JU,"

        VLC     = "K,P,S,T,F,X,C,CH,SH,SHCH,"

        VCN     = "B,G,GH,D,ZH,Z,"
        HRVW    = "A,E,Y,O,U,"
        KVW     = "U,I,JI,"

        SVW     = "U,I,JI,Y,"

RDWD    VRB     *V*                                     /F(END)
        CW      =
        NV      =

        V       *LTRT* ","

        LTSC    = LTRT ","

GETWN   LTSC    *WN/"1"*
        PERD    = "."
        EQUALS  (WN,PERD)                               /F(TORUP)

        WN      "A"                                     /S(AA)

        WN      "B"                                     /S(BB)

        WN      "V"                                     /S(VV)

        WN      "G"                                     /S(GG)

        WN      "GH"                                    /S(GH)

        WN      "D"                                     /S(DD)

        WN      "E"                                     /S(EE)

        WN      "JE"                                    /S(JE)

        WN      "ZH"                                    /S(ZH)

        WN      "Z"                                     /S(ZZ)

        WN      "Y"                                     /S(YY)

        WN      "I"                                     /S(II)

        WN      "JI"                                    /S(JI)

        WN      "J2"                                    /S(J2)
        WN      "K"                                              /S(KK)
        WN      "L"                                              /S(LL)
        WN      "M"                                              /S(MM)
        WN      "N"                                              /S(NN)
        WN      "O"                                              /S(OO)
        WN      "P"                                              /S(PP)
        WN      "R"                                              /S(RR)
        WN      "S"                                              /S(SS)
        WN      "T"                                              /S(TT)
        WN      "U"                                              /S(UU)
        WN      "F"                                              /S(FF)
        WN      "X"                                              /S(XX)
        WN      "C"                                              /S(CC)
        WN      "CH"                                             /S(CH)
        WN      "SH"                                             /S(SH)
        WN      "SHCH"                                           /S(SHCH)
        WN      "JU"                                             /S(JU)
        WN      "JA"                                             /S(JA)
        LESS    WN =                                             /(GETWN)
DIME    LESC    WN =
        XW      = XW ","
        XW      "," *SD/"2"* ",."
        ZW      = ZW WN "=" SD ","
        XW      ",+" = ","                                       /(GETWN)
STCK    OVTER   = "ENCA'                                         /(GETFOW)
TNCK    XW      = XW ","
        ..      "," *S1/"2"* "," *S2/"2"* ",+"
        ZW      = ZW WN "=" S1 S2 ","
```

```
          XW        ",+" = ","
          LTSC      WN =                                  /(GETWN)
STCX2     DVTER     = "TWCX"                              /(JE12)
STCX3     DVTER     = "TWCX"                              /(EE2)
GETSTR    STTT      = 0
          SDY       = "+" LTSC
DLWA      SDY       *WA/"1"* "." = "."
          WA        "+"                                   /S(ALDN)
          VWL       = VW
DLVW      VWL       *TVW* "," =                           /F(DLWA)
          EQUALS    (WA,TVW)                              /S(INCR)F(DLVW)
INCR      STTT      = STTT + "1"                          /(DLWA)
ALDN      V         *PRSTR/"25"* *STN/"1"*
          EQUALS    (STTT,STN)                            /S($ACGOBK)F($UNAGBK)
AA        ACGOBK    = "AA1"
          UNAGBK    = "AA2"                               /(GETSTR)
AA1       XW        = XW "A1,"                            /(DLWN)
AA2       XW        = XW "A4,"                            /(DLWN)
 JA       DVTER     = "DLWN"
          ACGOBK    = "JA1"
          UNAGBK    = "JA2"                               /(GETSTR)
JA1       NPRCN     = "JA11"
          FLHC      = "JA12"
          FLSC      = "JA13"
GTPRGB    PRGOBK    = "TSPRCN"
          ISPRCN    = "GETFLW"
          FLGOBK    = "TSTJ2"                             /(GETPRC)
JA2       ISPRCN    = "JA21"
          FLHC      = "JA22"
          FLSC      = "JA23"                              /(GTPRGB)
```

```
JA11      XW        = XW "I3,"                                 /(STCX)
JA12      XW        = XW "A2,"                                 /($DVTER)
JA13      XW        = XW "A3,"                                 /($DVTER)
JA21      XW        = XW "I4,"                                 /(STCX)
JA22      XW        = XW "A5,"                                 /($DVTER)
JA23      XW        = XW 'A6,"                                 /($DVTER)
EE        DVTER     = "DLWN"
          ACGOBK    = "EE1"
          UNAGBK    = "EE2"                                    /(GETSTR)
EE1       XW        = XW "E1,"                                 /(DLWN)
EE2       FLGOBK    = "ISFHGV"
          NFHV      = "EE21"
          ISFHV     = EE22"                                    /(GETFLW)
EE21      XW        = XW "E2,"                                 /($DVTER)
EE22      XW        = XW "Y2,"                                 /($DVTER)
JE        DVTER     = "DLWN"
          ACGOBK    = "JE1"
          UNAGBK    = "JE2"                                    /(GETSTR)
JE1       NPRCN     = "JE11"
          ISPRCN    = "JE12"
GTPRG1    PRGOBK    = "ISPRCN"                                 /(GETPRC)
JE11      XW        = XW "I3,"                                 /(STCX2)
JE12      XW        = XW "E1,"                                 /($DVTER)
JE2       NPRCN     = "JE21"
          ISPRCN    = "EE2"                                    /(GTPRG1)
JE21      XW        = XW "I4,"                                 /(STCX3)
II        ACGOBK    = "II1"
          UNAGBK    = "II2"                                    /(GETSTR)
```

```
II1     XW        = XW "I1,"                       /(DLWN)

II2     XW        = XW "I2,"                       /(DLWN)

JI      ACGOBK    = "JI1"

        UNAGBK    = "JI2"                          /(GETSTR)

JI1     XW        = XW "I3,I1,"                    /(TWCX)

JI2     XW        = XW "I4,I2,"                    /(TWCX)

YY      DVTER     = "DLWN"

        ACGOBK    = "YY1"

        UNAGBK    = "EE2"                          /(GETSTR)

YY1     XW        = XW "Y1,"

J2      FLGOBK    = "TFSTRV"

        ACGOBK    = "J21"

        UNAGBK    = "J22"                          /(GETFLW)

J21     XW        = XW "I3,"                       /(DLWN)

J22     XW        = XW "I4,"                       /(DLWN)

OO      ACGOBK    = "OO1"

        UNAGBK    = "OO2"                          /(GETSTR)

OO1     PRGOBK    = "TSTJ12"

        NSPC      = "OO11"

        SPRC      = "GETFLW"

        FLGOBK    = "TSTJ2"

        FLHC      = "OO12"

        FLSC      = "OO13"                         /(GETPRC)

OO11    XW        = XW "O1,"                       /(DLWN)

OO12    XW        = XW "O2,"                       /(DLWN)

OO13    XW        = XW "O3,"                       /(DLWN)

OO2     PRGOBK    = "TSTJ12"

        FLGOBK    = "TSTFRV"
        NSPC     = "0021"
        SPRC     = "0022"                              /(GETPRC)
0021    ISFRV    = "00211"
        NFRV     = "00212"                             /(GETFLW)
00211   XW       = XW "04,"                            /(DLWN)
00212   XW       = XW "05,"                            /(DLWN)
0022    ISFRV    = "00221"
        NFRV     ="00222"                              /(GETFLW)
00221   FLGOBK   = "TSTJ2"
        FLHC     = "002211"
        FLSC     = "002212"                            /(GETFLW)
002211  XW       = XW "08,"                            /(DLWN)
002212  XW       = XW "09,"                            /(DLWN)
00222   FLGOBK   = "TSTJ2"
        FLHC     = "002221"
        FLSC     = "002222"                            /(GETFLW)
002221  XW       = XW "06,"                            /(DLWN)
002222  XW       = XW "07,"                            /(DLWN)
UU      ACGOBK   = "UU1"
        UNAGBK   = "UU2"                               /(GETSTR)
UU1     FLGOBK   = "TSTJ2"
        FLHC     = "UU11"
        FLSC     = "UU12"                              /(GETFLW)
UU11 -  XW       = XW "U1,"                            /(DLWN)
UU12    XW       = XW "U2,"                            /(DLWN)
UU2     PRGOBK   = "DEFVWL"
        NVPR     = "UU21"
        VWPR     = "UU22"                              /(GETPRC)
```

```
UU[illegible]    XW        = XW "U5,"                          /(DL[illegible])
UU26     XW        = [illegible]                       /(DL[illegible])
JU       DVTER     = "[illegible]"
         [illegible]GOBK    = "JU[illegible]"
         [illegible]AGBK    = "JU[illegible]"
JU[illegible]     [illegible]RCN     = "JU[illegible]"
         [illegible]GC      = "JU[illegible]"
         [illegible]SC      = "JU[illegible]"
[illegible]        [illegible]GOBK    = "[illegible]"
         [illegible]FRCN    = "[illegible]"
         [illegible]GOBK    = "[illegible]"                     /(G[illegible])
JU2      [illegible]RCN     = "JU[illegible]"
         [illegible]GC      = "JU[illegible]"
         [illegible]SC      = "JU[illegible]"                   /(G[illegible])
JU[illegible]     XW        = XW "[illegible]"                  /(STCK)
JU[illegible]     XW        = XW "[illegible]"                  /($J[illegible])
JU[illegible]     XW        = XW "U4,"                          /($DVTER)
JU[illegible]     XW        = XW "U4,"                          /(STCK)
JU22     XW        = XW "U5,"                          /($DVTER)
JU23     XW        = XW "U7,"                          /($DVTER)
DB       SDVAR     = "D2,"
         FLVAR     = "B1,"                             /(DLGOGO)
VV       FLGOBK    = "[illegible]"
         IFWCN     = "VV1"
         EFWCN     = "VV2"                             /(G[illegible]FLM)
VV1      XW        = XW "U8,"                          /(DL[illegible])
VV2      FLGOBK    = "[illegible]FLM"
         FLGBLK    = "VV1"
```

```
          NFWBLK    = "VV3"                                   /(GETFLW)
VV3       FLGOBK    = "TNXHV"
          NNXHV     = "VV4"
          HVFLW     = "VV5"                                   /(GETFLW)
VV4       XW        = XW "V2,"                                /(DLWN)
          XW        = XW "V1,"                                /(DLWN)
GG        FLGOBK    = "TNXSV"
          NNXSV     = "GG1"
          SVFLW     = "GG2"                                   /(GETFLW)
GG1       FLGOBK    = "TFLBLK"
          FLWBLK    = "GG3"
          NFWBLK    = "GG4"                                   /(GETFLW)
GG2       XW        = XW "H2,"                                /(DLWN)
GG3       PRGOBK    = "TSPRCN"
          NFPRCN    = "GG5"
          ISPRCN    = "GG6"                                   /(GETPRC)
GG4       FLGOBK    = "TSFVCN"
          NFVLC     = "GG5"
          IFVLC     = "GG7"                                   /(GETFLW)
GG5       XW        = XW "H1,"                                /(DLWN)
GG6       XW        = XW "H3,"                                /(DLWN)
GG7       PRGOBK    = "DEFVWL"
          NVPR      = "GG5"
          VWPR      = "GG8"                                   /(GETPRC)
GG8       XW        = "X1,"                                   /(DLWN)
GH        SHVAR     = "GH2,"
```

```
        PLVAR   = "GH1,"                                  /(DDGOGO)
DD      FLGOBK  = "DD1"                                   /(GETFLW)
DD1     RLFL    "+Z"                                      /S(DZ1)
        RLFL    "+ZH"                                     /S(DZH1)
        SHVAR   = "D2,"
        PLVAR   = "D1,"                                   /(DDGOGO)
DZ1     WN      = "DZ"
        SHVAR   = "DH2,"
        PLVAR   = "DH1,"                                  /(DDGOGO)
DZH1    WN      = "DZH"
        SHVAR   = "J2,"
        PLVAR   = "J1,"                                   /(DDGOGO)
DDGOGO  DVTER   = "DLWN"
DDGOG1  FLGOBK  = "TNXSV"
        NNXSV   = "DD1"
        SVFLW   = "DD2"                                   /(GETFLW)
DD1     FLGOBK  = "TSCNS"
        NNXSMC  = "DD3"
        ISNXSM  = "DD4"                                   /(GETFLW)
DD2     XW      = XW SHVAR                                /($DVTER)
DD3     XW      = XW PLVAR                                /($DVTER)
DD4     NNXSV   = "DD3"                                   /(TNXSV)
ZH      FLGOBK  = "TFSORC"
        NFSORC  = "ZH3"
        NNXSV   = "ZH1"
        SVFLW   = "ZH2"                                   /(GETFLW)
ZH1     XW      = XW "Z1,"                                /(DLWN)
```

```
ZH2     XW       = XW "Z2,"                    /(DLWN)
ZH3     SHVAR    = "ZH2,"
        PLVAR    = "ZH1,"                      /(DDGOGO)
ZZ      PRGOBK   = "TSPRBK"
        IPRBK    = "ZZ1"
        NPRBK    = "ZZ2"                       /(GETPRC)
ZZ1     FLGOBK   = "TFVLPC"
        PLMOD    = "SH1,"
        NFPC     = "ZZ12"                      /(GETFLW)
ZZ12    PLMOD    = "ZH1,"
        NFPC     = "ZZ13"                      /(TSFPLC)
ZZ13    RLFL     "+S"   <                      /F(ZZ14)
        XW       = XW "S1,"                    /(DLWN)
ZZ14    SHVAR    = "Z2,"
        PLVAR    = "Z1,"                       /(DDGOGO)
ZZ2     PLMOD    = "ZH1,"
        NFPC     = "ZZ14"
        FLGOBK   = "TSFPLC"                    /(GETFLW)
LL      SHVAR    = "L2,"
        PLVAR    = "L1,"                       /(DDGOGO)
MM      SHVAR    = "M2,"
        PLVAR    = "M1,"                       /(DDGOGO)
NN      SHVAR    = "N2,"
        PLVAR    = "N1,"                       /(DDGOGO)
RR      SHVAR    = "R2,"
        PLVAR    = "R1,"                       /(DDGOGO)
```

```
KK       VDVAR    = "G1,"
         PLVAR    = "K1,"
         SHVAR    = "K2,"
KKGO     DVTER    = "DLWN"
         FLGOBK   = "IFVCN"
         NFVCN    = "DDGOGO"
KKØ      IFVCN    = "KK1"                                      /(GETFLW)
KK1      XW       = XW VDVAR                                   /($DVTER)
KKG01    FLGOBK   = "IFVCN"
         NFVCN    = "DDGOG1"                                   /(KKØ)
PP       VDVAR    = "B1,"
         PLVAR    = "P1,"
         SHVAR    = "P2,"                                      /(KKGO)
SS       FLGOBK   = "IFVLPC"
         NFPC     = "SS1"
         PLMOD    = "SH1,"                                     /(GETFLW)
SS1      PLMOD    = "ZH1,"
         NFPC     = "SS2"                                      /(TSFPLC)
SS2      PRGOBK   = "SS21"                                     /(GETPRC)
SS21     RLPR     = "TJ+"                                      /F(SS3)
         XW       = XW "C2,"                                   /(DLWN)
SS3      VDVAR    = "Z1,"
         PLVAR    = "S1,"
         SHVAR    = "S2,"                                      /(KKGO)
TT       FLGOBK   = "IFVLPC"
         PLMOD    = "CH1,"
```

```
          NFPC      = "TFCONL"
          NFSORC    = "TT3"
          NNXSV     = "TT1"
          SVFLW     = "TT2"                                  /(GETFLW)
TT1       XW        = XW "C1,"                               /(DLWN)
TT2       XW        = XW "C2,"                               /(DLWN)
TT3       FLGOBK    = "TT31"                                 /(GETFLW)
TT31      RIFL      "+JSJA"                                  /(S(TT2)F(TT4)
TT4       VDVAR     = "D1,"
          PLVAR     = "T1,"
          SHVAR     = "T2,"                                  /(KKGO)
FF        VDVAR     = "V1,"
          PLVAR     = "F1,"
          SHVAR     = "F2,"                                  /(KKGO)

XX        VDVAR     = "GH1,"
          PLVAR     = "X1,"
          SHVAR     = "X2,"                                  /(KKGO)
CC        VDVAR     = "DH1,"
          PLVAR     = "C1,"
          SHVAR     = "C2,"                                  /(KKGO)
CH        FLGOBK    = "TFSORC"
          NFSORC    = "CH3"
          NNXSV     = "CH1."
          SVFLW     = "CH2"                                  /(GETFLW)
```

```
CH1       XW        = XW "C1,"                              /(DLWN)
CH2       XW        = XW "C2,"                              /(DLWN)
CH3       VDVAR     = "J1,"
          PLVAR     = "CH1,"
          SHVAR     = "CH2,"                                /(KKGO)
SH        PRGOBK    = "SH1"                                 /(GETPRC)
SH1       RLPR      = "SJ+"                                 /F(SH2)
          XW        = XW "CH1,"                             /(DLWN)
SH2       FLGOBK    = "ITFSORC"
          NFSORC    = "SH5"
          NMKSV     = "SH3"
          SVFLW     = "SH4"                                 /(GETFLW)
SH3       XW        = XW "S1,"                              /(DLWN)
SH4       XW        = XW "S2,"                              /(DLWN)
SH5       VDVAR     = "ZH1,"
          PLVAR     = "SH1,"
          SHVAR     = "SH2,"                                /(KKGO)
SHCH      VDVAR     = "ZH1,J1,"
          PLVAR     = "SH1,CH1,"
          SHVAR     = "SH2,CH2,"
          DVTER     = 'TWCX"                                /(KKGO1)
GETPRC    LTSC      *PRDM* "."
          TSPR      = "+" LTPT
          TSPR      *RLPR* PRDM
          RLPR      = RLPR "+"                              /($PRGOBK)
TSPRBK    RLPR      "++"                        /S($IPRBK)F($NPRBK)
```

```
TSPRCN    CNST     = CNS                                   /F($NFRCN)
BRCN      CNST     *TCNS* "," =                            /F($NFRCN)
          RLP      TCNS "+"                                /S($ISPRCN)F(BRCN)
TSTJ12    RLPR     "J1+"                                   /S($SPRC)
          RLPR     "J2+"                                   /S($SPRC)F($NSPC)
DEFVWL    VWL      = VW
GPVW      VWL      *TVW* "," =                             /F($NVPR)
          RLPR     TVW "+"                                 /S($VWPR)F(GPVW)
GETFLM    LSTC     *FLDM* "."
          FLCM     = "+" FLDM "+"
          FLDM     WN =
          RLFL     = FLDM                                  /($FLGOBK)
TFBLK     RLFL     "++"                                    /S($FLWBLK)F($NFWBL
TSFWCN    CNST     = CNS
BRFCN     CNST     *TCNS* "," =                            /F($NFWCN)
          RLFL     "+" TCNS                                /S($IFWCN)F(BRFCN)
TSTJ2     RLFL     "+J2"                                   /S($FLSC)
          VWL      = VW
GFVW      VWL      *TVW* "," =                             /F(AVWG)
          RLFL     "+" TVW                                 /S(VWFL)F(GFVW)
AVWG      RLFL     "++"                                    /S($FLHC)
          SVWG     = "TSTDBC"
          RLFL     "+" *RLON/"1"* *RRST* "+"               /F($FLHC)
          RRST     = "+" RRST "+"
          SVWL     = SVW "J1,"
GFSV      SVWL     *TSVW* "," =                            /F($SVWG)
```

```
         RLFL     "+" TSVW                       /S($FLSC)F(GFSW)
TSTDBC   RRST     "+" RLON                       /F($FLHC)
         RRST     "+" RLON = "+"
         RLFL     = RRST                         /(VWFL)
TSCNS    RLFL     "+" WN                         /F($NNXSMC)
         RLFL     "+" WN = "+"                   /($ISNNXSM)
TSFVCN   VLCS     = VLC
GTFVLC   VLCS     *TVLC* ","                     /F($NFVLC)
         RLFL     "+" TVLC                       /S($IFVLC)F(GTFVLC)
TFVCN    VCNS     = VCN


GTVCN    VCN      *TVCN* ","                     /F($NFVCN)
         RLFL     "+" TVCN                       /S($IFVCN)F(GTVCN)
TSFPLC   RLFL     "+ZH"                          /S(IFPC)
         RLFL     "+DZH"                         /S(IFPC)
TFVLPC   RLFL     "+SH"                          /S(IFPC)
         RLFL     "+CH"                          /S(IFPC)
         RLFL     "+SHCH"                        /S(IFPC)F($NFPC)
IFPC     XW       = XW PLMOD                     /(DLWN)
TFSORC   RLFL     "+S"                           /S(TSNXS)
TFCONL   RLFL     "+C"                           /F($NFSORC)
         RLFL     "+C" = "+"                     /(TNXSV)
TSNXS    RLFL     "+S" = "+"                     /(TNXSV)
TNXHV    HVWL     = HRVW
BRHV     HVWL     *THVW* ","                     /F($NNXHV)
         RLFL     "+" THVW                       /S($HVFLW)F(BRHV)
```

```
TSTFRV    VWL       = VW
GTAVL     VWL       *TVW* "," =                          /F(OUTV)
          RLFL      "+" TVW                              /S(ISFW)F(GTAVL)
OUTV      RLFL      "++"                                 /S($NFRV)
          RLFL      "+" *RLON/"1"* *RRST* "+"            /F($NFRV)
          RLFL      = "+" RRST "+"                       /(TSTFRV)
ISVW      RVWL      = RVW
GTRV      RVWL      *TRVW* "," =                         /F($NFRV)
          TVW       TRVW                                 /S($ISFRV)F(GTRV)
TNXSV     SVWL      = SVW "J1,"
BRSV      SVWL      *TSVW* ","                           /F($NMXSV)
          RLFL      "+" TSVW                             /S($SVFLW)F(BRSV)
TSFHGV    VWL       = VW
GTNVW     VWL       *TVW* "," =                          /F(NMRV)
          RLFL      "+" TVW                              /S(ISFV)F(GTNVW)
NMRV      RLFL      "++"                                 /S($NFHV)
          TLFL      "+" *RLON/"1"* *RRST* "+"            /F($NFHV)
          RLFL      = "+" RRST "+"                       /(TSFHGV)
ISFV      HVWL      = HVW
GTHV      HVWL      *THVW* "," =                         /F($NFHV)
          TVW       THVW                                 /S($ISFHV)F(GTHV)
TFSTRV    VWL       = VW
GTSTV     VWL       *TVW* ","                            /F(SUMAGBK)
          RLFL      "+" TVW                              /S(GETSTR)F(GTSTV)
```

```
TOMUP    XW       = XW ","
         NSDS     *SNLST* " "
GTFSD    XW       *FSD* "," =                                  /F(NOMOXW)
         SNDLT    = SNLST ","
GTTSD    SNDLT    *TSD* "," *SNM* "," =                        /F(GTFSD)
         EQUALS   (FSD,TSD)                                    /F(GTTSD)
         SNM      "1"                                          /S(UPSREI)
         NW       = NW + "1"                                   /(GTFSD)
UPSREI   RE       = RE + "1"                                   /(GTFSD)
NOMOXW   ZW       = ZW ","
         NSLTS    *SLLST* " "
GTFSL    ZW       *FSL* "," =                                  /F(NOMOZW)
         SDLLT    = SLLST ","
GTTSL    SDLLT    *TSL* "," *SLM* "," =                        /F(GTFSL)
         EQUALS   (FSL,TSL)                                    /F(GTTSL)
         SLM      "1"                                          /S(UPSLRE)
         NE       = NE + "1"                                   /(GTFSL)
UPSREI   RI       = RI + "1"                                   /(GTFSL)
NOMOZW   NWV      = V "," RE "," NW "," RI "," NE
         RTG      = NWV
         RE       =
         NW       =
         RI       =
         NE       =                                            /(RDVB)
END      BEGIN
```

Appendix 7

```
BEGIN     NVB       =
READIN    SYSPIT    *V/"72*

          V         "SJA "                 /S(RFL)
P1        V         "YTJ "                 /F(EP1)
          V         "PYTJ "                /S(LP1)
          V         "BYTJ "                /S(LP1)
          V         "VYTJ "                /S(LP1)
          V         "MYTJ "                /S(LP1)
          V         "FYTJ "                /S(LP1)

          V         "RYTJ "                /S(SP1)
          V         "LYTJ "                /S(SP1)
          V         "NYTJ "                /S(SP1)
          V         "CHYTJ "               /S(PP1)
          V         "ZHYTJ "               /S(PP1)
          V         "SHYTJ "               /S(PP1)
          V         "SHCHYTJ "             /S(PP1)

          V         "TYTJ "                /S(TP1)
          V         "DYTJ "                /S(DP1)
          V         "SYTJ "                /S(SSP1)

          NPR1      = "AZHU,DCVC"          /(P1S)
SSP1      NPR1      = "ASHU,DCVC"          /(P1S)
DP1       V         "ZDYTJ "               /S(ZDP1)
          NPR1      = "ADZHU,DCVC"         /(P1S)
```

```
ZDP1     NPR1     = "AZHDZHU,DCCVC"      /(P1S)
TP1      V        "STYTJ "               /S(STP1)
         NPR1     = "ACHU,DCVC"          /(P1S)
STP1     NPR1     = "ASHCHU,DCCVC"       /(P1S)
PP1      NPR1     = "AU,DVC"             /(P1S)
SP1      NPR1     = "AJU,DVC"            /(P1S)
LP1      NPR1     = "ALJU,DVC"           /(P1S)

P1S      V        *X/"31"* "1"           /F(XP1)
         NPR1     = NPR1 ",S1"
XP1      NVB      = NPR1                 /(P3)

EP1      V        "E "                   /F(JEP1)
         V        "ORE "                 /S(EP11)
         V        "LE "                  /S(EP11)
         NPR1     = "AU,DV"              /(P1S)
EP11     NPR1     = "AJU,DV"             /(P1S)

JEP1     V        "JE "                  /S(EP11)
         V        "JITJ "                /S(SP1)
         NPR1     = "AM,DCC"             /(XP1)

P3       V        "YTJ "                 /F(EP3)
         V        "BYTJ "                /S(LP3)
         V        "PYTJ "                /S(LP3)
```

```
        V           "VYTJ "               /S(LP3)
        V           "MYTJ "               /S(LP3)
        V           "FYTJ "               /S(SL3)
        V           "CHYTJ "              /S(PP3)
        V           "ZHYTJ "              /S(PP3)
        V           "SHYTJ "              /S(PP3)
        V           "SHCHYTJ "            /S(PP3)
P31     NPR3        = "AJATJ,DVC"
SP3     NVB         = NVB "+" NPR3        /(PL)
LP3     NPR3        = "ALJATJ,DVC"        /(SP3)
PP3     NPR3        = "AATJ,DVC"          /(SP3)
EP3     V           "E "                  /F(JEP3)
        V           "ORE "                /S(EP31)
        V           "LE "                 /S(EP31)
        NPR3        = "AUTJ,DV"           /(SP3)
EP31    NPR3        = "AJUTJ,DV"          /(SP3)
JEP3    V           "JE "                 /S(EP31)
        V           "JITJ "               /S(P31)
        V           "ASTJ "               /S(DAP3)
        NPR3        = "ADJATJ,DCC"        /(SP3)
DAP3    NPR3        = "ADUTJ,DCC"         /(SP3)

PL      V           "TJ "                 /F(EPL)
        V           "STJ "                /S(CPL)
        NP1P        = "AMO,DVC"
```

```
PLA      V        *Y/"25")* "1"          /F(SPL)
         NP1P     = NP1P ",S1"
SPL      NVB      = NVB "+" NP1P         /(I)
CP1P     NP1P     = "AMO,DCC"            /(PLA)
EP1P     NP1P     = "AMO,DV"             /(PLA)

I        V        "YTJ "                 /F(EI)
I1       V        *Z/"27"* "1"           /S(IBD)
IS       V        *XX/"31"* "1"          /S(I2)
         V        XX "2"                 /S(I2)
         V        *XY/"25"* "1"          /F(I3)
IBD      NIS      = "AY,DVC"             /(SI)
I2       NIS      = "AY,DVC,S1"          /(SI)
I3       V        "TYTJ "                /S(I4)
         V        "NYTJ "                /S(I4)
         V        "LYTJ "                /S(I4)
         V        "DYTJ "                /S(I4)
         V        "SYTJ "                /S(I4)
         V        "ZYTJ "                /S(I4)
         NIS      = "DVC"                /(SI)
I4       NIS      = "AJ1,DVC"
SI       NVB      = NVB "+" NIS          /(PM)

EI       V        "E "                   /F(JI)
         V        *A/"27"* "1"           /S(EI1)
```

```
          V      A "2"               /S(EI4)
          V      *B/"31"* "1"        /S(EI2)
          V      B "2"               /S(EI2)
          V      *C/"25"* "1"        /F(EI3)
EI1       NIS    = "AY,DV"           /(SI)
EI2       NIS    = "AY,DV,S1"        /(SI)
EI3       V      "NE "               /S(EI5)
          V      "LE "               /S(EI5)
          V      "TE "               /S(EI5)
          V      "DE "               /S(EI5)
          V      "SE "               /S(EI5)
          V      "ZE "               /S(EI5)
          NIS    = "DV"              /(SI)
EI5       NIS    = "AJ1,DV"          /(SI)
EI4       NIS    = "DCV"             /(SI)

JI        V      "JE "               /F(II)
          V      "LJE "              /S(LJI)
          V      "*JE "              /S(SJI)
          V      *D/"25"* "1"        /S(JI1)
          NIS    = "AJ2,DV"          /(SI)
JI1       V      "AJE "              /S(JI2)
          NIS    = "AJ2,DV,S-1"      /(SI)
JI2       NIS    = "AVAJ,DV"         /(SI)
SJI       NIS    = "AYJ,D*V"
SJI2      V      *E/"31"* "1"        /F(SJI1)
```

	NIS	= NIS ",S1"	
SJI1	V	*F/"34"* "1"	/F(SI)
	NIS	= NIS ",OPLV"	/(SI)
LJI	NIS	= "AYJ,DCV"	
	V	*G/"34"* "2"	/F(SJI2)
	NIS	= NIS ",OPFVPLV"	/(SJI2)
II	V	"JITJ "	/F(CI)
	V	*H/"27"* "1"	/S(II1)
	NIS	= "AJ2,DVC"	
II2	V	*HI/"25"* "1"	/F(SI)
	NIS	= NIS ",S-1"	/(SI)
II1	NIS	= "AIJ,DVVC"	/(II2)
CI	V	"ASTJ "	/S(DAI)
	NIS	= "AZH,DCC"	/(SI)
DAI	NIS	= "AJ2,DCC"	/(SI)
PM	V	"YTJ "	/F(EM)
	V	*I/"28"* "1"	/S(M2)
	V	I "2"	/S(M3)
	V	I "3"	/S(M4)
	V	I "4"	/S(M5)
PM1	NM	= "AV,DC"	
MS	V	*J/"31"* "1"	/F(SM)
MS1	NM	= NM ",S1"	/(SM)
M2	NM	= "AIV,DVC"	/(MS)
SM	NVB	= NVB "+" NM	/(PF)

```
M3        NM     = "AAV,DVC"         /(MS)
M4        NM     = "AG,DCVC"
AM        V      *K/"25"* "1"        /F(SM)
          NM     = NM ",S-1"         /(SM)
M5        NM     = "AKAV,DCVC"       /(MS)

EM        V      "E "                /F(JEM)
          V      "ZHE "              /F(SHM)
          V      *I/"28"* "1"        /S(M6)
          V      I "2"               /S(M7)
          V      I "3"               /S(M8)
          NM     = "AG,DCV"          /(AM)
M6        NM     = "AIG,DVCV"        /(AM)
M7        NM     = "AGAV,DCV"        /(MS)
M8        NM     = "AZAV,DCV"        /(MS)

SHM       V      "SHE "              /F(CHM)
          V      *I/"28"* "1"        /S(M9)
SHM1      NM     = "AXAV,DCV"        /(MS)
M9        NM     = "ASAV,DCV"        /(MS)

CHM       V      "CHE "              /F(SCM)
          V      *I/"28"* "1"        /S(HM2)
          V      I "2"               /S(HM3)
          V      I "3"               /S(HM4)
          V      I "4"               /S(HM5)
          NM     = "AK,DCV"          /(AM)
```

```
HM2     NM      = "AIK,DVCV"        /(AM)
HM3     NM      = "AKAV,DCV"        /(MS)
HM4     NM      = "ATAV,DCV"        /(MS)
HM5     NM      = "ATIV,DCV,S1"     /(SM)

SCM     V       "SHCHE "            /F(NEM)
        V       *I/"28"* "1"        /S(SCHM)
        V       = "ASKAV,DCV"       /(MS)
SCHM    NM      = "ASTAV,DCV"       /(MS)

NEM     V       "NE "               /F(MM)
        V       *I/"28"* "1"        /S(NM1)
        V       I "2"               /S(NM2)
        V       I "3"               /S(NM2)
        V       I "4"               /S(NM3)
        V       I "5"               /S(NM4)
        V       I "6"               /S(NM5)
        V       I "7"               /S(NM6)
        NM      = "AUV,DV"          /(MS)
NM1     NM      = "AAV,DV"          /(NM7)
NM2     NM      = "DCV"             /(AM)
NM3     NM      = "AV,DCV"          /(AM)
NM4     NM      = "AAV,DCV"
```

```
NMO        V      *L/"34"* "1"             /F(MS)
NMG        NM     = NM ",OPLV"             /(MS)
NM5        NM     = "A*JAV,DCV"            /(NMO)
NM6        NM     = "AJAV,DCV"
           V      *M/"34"* "2"             /F(NMO)
           NM     = NM ",CPFCS"            /(NMG)
NM7        V      *N/"36"* "1"             /F(NM71)
           NM     = NM ",IAPLCI"
NM71       V      *LI/"34"* "1"            /F(MS)
           NM     = NM ",OSLV,CSFCG"       /(SM)
MM         V      "ME "                    /F(RM)
           V      *I/"28"* "1"             /S(MM1)
           V      I "3"                    /S(MM2)
           V      I "3"                    /S(MM3)F(NM4)
MM1        NM     = "ANJAV,DCV"            /(MS)
MM2        NM     = "AJAV,DVCV"            /(SM)
MM3        NM     = "AUV,DCV"              /(NMO)

RM         V      "RE "                    /F(LM)
           V      *I/"28"* "1"             /S(RM1)
           V      I "2"                    /S(RM2)
           V      "ERE "                   /S(RM3)
           NM     = "DV,IBSLCE"
           V      *M/"34"* "2"             /F(ROM)
           NM     = NM ",CPFCS"            /(RMG)
```

ROM	V	*L/"34"* "1'	/F(AM)
RMG	NM	= NM ",OPLV"	
	V	*J/"31"* "1"	/S(MS1)F(AM)
RM3	NM	= "DV "	/(AM)
RM1	NM	= "AAV,DV"	
	V	*M/"34"* "1"	/F(MS)
	NM	= NM ",OSLV"	
	V	*N/"36"* "1"	/F(RM1)
	NM	= NM ",IAPLCI"	/(SM)
RM11	V	N "2"	/F(RM12)
	NM	= NM ",IAPLVVI"	/(SM)
RM12	V	*O/"35"* "1"	/F(MS)
	NM	= NM ",CPLVO"	/(SM)
RM2	NM	= "AOV,DV"	/(MS)
LM	V	"LE "	/F(DM)
	V	*I/"28"* "1"	/S(NM4)
	V	I "2"	/S(LM1)
	NM	= "AAV,DV,CSFCS"	
	V	*L/"34"* "1"	/F(MS)
	NM	= "AAV,DV,OSLCV"	/(MS)
LM1	V	"ELE "	/F(RM2)

	NM	= "AOV,DV,CSLVO"	/(MS)
DM	V	"DE "	/F(TM)
	V	*I/"28"* "1"	/S(DM1)
	V	I "2"	/S(DM1)
	V	I "3"	/S(RM1)
	V	I "4"	/S(SHM1)
	V	I "5"	/S(DM2)
	V	I "6"	/S(DM3)
DM0	NM	= "AV,DCV"	/(AM)
DM1	NM	= "AIV,DCV"	/(AM)
DM2	NM	= "ASHOV,DCV"	/(MS)
DM3	NM	= "AIV,DV"	/(MS)
TM	V	"TE "	/F(VM)
	V	*I/"28"* "1"	/S(DM1)
	V	*O/"35"* "1"	/F(DM0)
	NM	= "AV.DCV,CSLVI"	/(AM)
VM	V	*I/"28"* "1"	/S(RM1)
	V	I "2"	/S(DM3)
	V	*O/"35"* "1"	/F(RM3)
	NM	= "DV,CSLVI"	/(AM)
JEM	V	"JE "	/F(IM)
	V	"LJE "	/S(JM2)
	V	"*JE "	/S(JM3)
	V	"UJE "	/S(JM4)

	V	*I/"28"* "1"	/S(JM5)
	V	*K/"25"* "1"	/S(JM1)
JM0	NM	= "AV,DV"	/(AM)
JM1	V	"AJE "	/F(JM0)
JM11	NM	= "AVAV,DV"	/(SM)
JM2	NM	= "AYV,DCV"	
JM21	V	*L/"34"* "1"	/S(NMG)
	V	L "2"	/F(SM)
	NM	= NM ",OPFVPLV"	/(SM)
JM3	NM	= "AYV,D*V"	/(JM21)
JM4	V	*I/"28"* "1"	/S(JM0)
	V	*K/"25"* "2"	/F(JM11)
	NM	= "AVAV,DV,S1"	/(SM)
JM5	NM	= "AJAV,DV"	/(MS)
IM	V	"JITJ "	/F(CM)
	V	*I/"28"* "1"	/F(PM1)
	NM	= "AJAV,DVC"	/(MS)
CM	NM	= "AV,DCC"	/(SM)
PF	V	"YTJ "	/F(EP)
	V	*P/"28"* "1"	/S(F1)
	V	P "2"	/S(F2)
	V	P "3"	/S(F3)
	V	P "4"	/S(F4)
PF1	NF	= "ALA,DC"	

FS	V	*Q/"31"* "1"	/F(SF)
FS1	NF	= NF ",S1"	
SF	NVB	= NVB "+" NF	/NTV)
F1	NF	= "AILA,DVC"	/(FS)
F2	NF	= "AALA,DVC"	/(FS)
F3	NF	= "AGLA,DCVC"	
FA	V	*R/"25"* "1"	/F(FS)
FA1	NF	= NF ",S-1"	/(SF)
F4	NF	= "AKALA,DCVC"	/(FS)
EF	V	"E "	/F(JF)
	V	"ZHE "	/F(SHF)
	V	*P/"28"* "2"	/S(ZHF1)
	V	P "3"	/S(ZHF2)
	NF	= "AGLA,DCV"	
EF1	V	*S/"31"* "2"	/S(FS1)
	V	S "3"	/S(FA1)
	V	S "4"	/S(FS1)F(FS)
ZHF1	NF	= "AGALA,DCV"	/(FS)
ZHF2	NF	= "AZALA,DCV"	/(FS)
SHF	V	"SHE "	/F(CHF)
	V	*P/"28"* "1"	/S(SHF1)
SHF0	NF	= "AXALA,DCV"	/(FS)
SHF1	NF	= "ASALA,DCV"	/(FS)
CHF	V	"CHE "	/F(SCF)
	V	*P/"28"* "1"	/S(CHF1)

```
          V     P "2"                   /S(CHF2)
          V     P "3"                   /S(CHF3)
          NF    = "AKLA,DCV"            /(EF1)
CHF1      NF    = "AKALA,DCV"           /(FS)
CHF2      NF    = "ATALA,DCV"           /(FS)
CHF3      NF    = "ATILA,DCV,S1"        /(SF)
SCF       V     "SHCHE "                /F(NEF)
          V     *P/"28"* "1"            /S(SCF1)
          NF    = "ASKALA,DCV"          /(FS)
SCF1      NF    = "ASTALA,DCV"          /(FS)

NEF       V     "NE "                   /F(MF)
          V     *P/"28* "1"             /S(NF1)
          V     P "2"                   /S(NF2)
          V     P "3"                   /S(NF2)
          V     P "4"                   /S(NF2)
          V     P "5"                   /S(NF3)
          V     P "6"                   /S(NF4)
          V     P "7"                   /S(NF5)
          NF    = "AULA,DV"             /(FS)
NF1       NF    = "AALA,DV"
          V     *T/"36"* "1"            /F(NF11)
          NF    = NF ",IAPLCI"
NF11      V     *U/"34* "1"             /F(FS)
          NF    = NF ",OSLV,CSFCG"      /(SF)
```

```
NF2       NF     = "ALA,DCV"          /(FS)
NF3       NF     = "AALA,DCV"
NF32      V      *U/"34"* "1"         /F(NF31)
NF321     NF     = NF ",OPLV"
NF31      V      *QQ/"31"* "4"        /S(FS1)F(FS)
NF4       NF     = "A*JALA,DCV"       /(NF32)
NF5       NF     = "AJALA,DCV"
NF52      V      *W/"34"* "2"         /F(NF51)
          NF     = NF ",CPFCS"        /(NF321)
NF51      V      *AA/"32" "4"         /F(NF32)
          NF     = NF ",S1"           /(NF321)

MF        V      "ME "                /F(RF)
          V      *P/"28"* "1"         /S(MF1)
          V      P "2"                /S(MF2)
          V      P "3"                /S(MF3)F(NF3)
MF1       NF     = "ANJALA,DCV"       /(NF51)
MF2       NF     = "AJALA,DVCV"       /(FS)
MF3       NF     = "AULA,DCV"         /(NF32)
RF        V      "RE "                /F(LF)
          V      *P/"28"* "1"         /S(RF1)
          V      P "2"                /S(RF2)
          NF     = "ALA,DV"
          V      "ERE "               /S(FA)
          NF     = NF ",IBSLCE"
```

	V	*BB/"25"* "1"	/F(NF32)
	NF	= NF ",S-1"	/(NF52)
RF1	NF	= "AALA,DV"	
	V	*U/"34"* "1"	/F(FS)
	NF	= NF ",OSLV"	
	V	*T/"36"* "1"	/F(RF11)
	NF	= NF ",IAPLCI"	/(SF)
RF11	V	T "2"	/F(RF12)
	NF	= NF ",IAPLVVI"	/(SF)
RF12	V	*CC/"35"* "1"	/F(FS)
	NF	= NF ",CPLVO"	/(SF)
RF2	NF	= "AOLA,DV"	/(FS)
LF	V	"LE "	/F(DF)
	V	*P/"28" "1"	/S(NF3)
	V	P "2"	/S(LFL)
	NF	= "AALA,DV,CSFCS"	
	V	*U/"34"* "1"	/F(FS)
	NF	= "AALA,DV,OSLCV"	/(FS)
LF1	V	"ELE "	/F(RF2)
	NF	= "AOLA,DV,CSLVO"	/(FS)
DF	V	"DE "	/F(TF)
	V	*P/"28"* "2"	/S(DF2)
	V	P "3"	/S(NF1)

	V	P "4"	/S(SHF0)
	V	P "5"	/S(DF3)
	V	P "6"	/S(DF4)
DF1	NF	= "ALA,DCV"	
DF11	V	*DD/"31* "3"	/S(FA1)F(NF31)
DF2	NF	= "AILA,DVCV"	/(FS)
DF3	NF	= "ASHLA,DCV"	/(FS)
DF4	NF	= "AILA,DV"	/(FS)
TF	V	"TE "	/S(DF1)
VF	V	*P/"28"* "1"	/S(RF1)
	V	P "2"	/S(DF4)
	V	P "3"	/S(DF1)
	NF	= "ALA,DV"	/(DF11)
JF	V	"JE "	/F(TF)
	V	"LJE "	/S(LJF)
	V	"*JE "	/S(SJF)
	V	"UJE "	/S(UJF)
	V	*P/"28"* "1"	/S(JF2)
	V	*R/"25"* "1"	/S(JF1)
JF0	NF	= "ALA,DV"	/(SF)
JF1	V	"AJE "	/F(JF0)
JF11	NF	= "AVALA,DV"	/(SF)
LJF	NF	= "AYLA,DCV"	

```
LJF1     V      *FF/"34"* "1"          /S(NF51)
         V      FF "2"                 /F(NF31)
         NF     = NF ",OPFVPLV"        /(NF31)

SJF      NF     = "AYLA,D*V"           /(LJF1)
UJF      V      *P/"28"* "1"           /S(JF0)
         V      *GG/"25"* "2"          /F(JF11)
         NF     = "AVALA,DV,S1"        /(SF)
JF2      NF     = "AJALA,DV"           /(FS)

IF       V      "JITJ "                /F(CF)
         V      *P/"28"* "1"           /F(PF1)
         NF     = "AJALA,DVC"          /(FS)

CF       NF     = "ALA,DCC"            /(NF31)

NTV      V      "YTJ "                 /F(EN)
         V      *HH/"28"* "1"          /S(N1)
         V      HH "2"                 /S(N2)
         V      HH "3"                 /S(N3)
         V      HH "4"                 /S(N4)
NTV1     NN     = "ATY,DC"
NS       V      *II/"31"* "1"          /F(SN)
NS1      NN     = NN ",S1"
SN       NVB    = NVB "+" NN "."       /(FINAL)
N1       NN     = "AITY,DVC"           /(NS)
```

N2	NN	= "AATY,DVC"	/(NS)
N3	NN	= "AGTY,DCVC"	
NA	V	*JJ/"25"* "1"	/F(NS)
NA1	NN	= ",S-1"	/(SN)
N4	NN	= "AKATY,DCVC"	/(NS)
EN	V	"E "	/F(JN)
	V	"ZHE "	/F(SHN)
	V	*HH/"28"* "1"	/S(ZHN1)
	V	HH "2"	/S(ZHN2)
	NN	= "AGTY,DCV"	
EN1	V	*KK/"31"* "2"	/S(NS1)
	V	KK "3"	/S(NA1)
	V	KK "4"	/S(NS1)F(NS)
ZHN1	NN	= "AGATY,DCV"	/(NS)
ZHN2	NN	= "AZATY,DCV"	/(NS)
SHN	V	"SHE "	/F(CHN)
	V	*HH/"28"* "1"	/S(SHN1)
SHN0	NN	= "AXATY,DCV"	/(NS)
SHN1	NN	= "ASATY,DCV"	/(NS)
CHN	V	"CHE "	/F(SCN)
	V	*HH/"28"* "1"	/S(CHN1)

```
          V     HH "2"                  /S(CHN2)
          V     HH "3"                  /S(CHN3)
          NN    = "AKTY,DCV"            /(EN1)
CHN1      NN    = "AKATY,DCV"           /(NS)
CHN2      NN    = "ATATY,DCV"           /(NS)
CHN3      NN    = "ATITY,DCV,S1"        /(SN)
SCN       V     "SHCHE "                /F(NEN)
          V     *HH/"28"* "1"           /S(CN1)
          NN    = "ASKATY,DCV"          /(NS)
SCN1      NN    = "ASTATY,DCV"          /(NS)

NEN       V     "NE "                   /F(MN)
          V     *HH/"28"* "1"           /S(NN1)
          V     HH "2"                  /S(NN2)
          V     HH "4"                  /S(NN3)
          V     HH "5"                  /S(NN4)
          V     HH "6"                  /S(NN5)
          V     HH "7"                  /S(NN6)
          NN    = "AUTY,DV"             /(NS)
NN1       NN    = "AATY,DV"
          V     *LL/"36"* "1"           /F(NN11)
          NN    = NN ",IAPLCI"
NN11      V     *PP/"34"* "1"           /F(NS)
          NN    = NN ",OSLV,CSFCG"      /(SN)
NN2       NN    = "ATY,DCV"             /(NS)
```

NN3	V	*RR/"29"* "1"	/F(NN2)
	NN	= "ASTY,DCV"	/(SN)
NN4	NN	= "AATY,DCV"	
NN42	V	*SS/"34"* "1"	/F(NS)
NN421	NN	= NN ",OPLV"	/(NS)
NN5	V	*RR/"29"* "1"	/S(NN51)
	NN	= "A*JATY,DCV"	/(SN)
NN51	NN	= "A*JASTY,DCV"	
	V	*TT/"31"* "4"	/F(NN42)
	NN	= NN ",S1"	/(NN42)
NN6	NN	= "AJATY,DCV"	
NN62	V	*UU/"34"* "2"	/F(NN42)
	NN	= NN ",CPFCS"	/(NN421)
MN	V	"ME "	/F(RN)
	V	*HH/"28"* "1"	/S(MN1)
	V	HH "2"	/S(MN2)
	V	HH "3"	/S(MN3)F(NN4)
MN1	NN	= "ANJATY,DCV"	/(NS)
MN2	NN	= "AJATY,DVCV"	/(SN)
MN3	NN	= "AUTY,DCV"	/(NN42)
RN	V	"RE "	/F(LN)
	V	*HH/"28"* "1"	/S(RN1)
	V	HH "2"	/S(REN2)
	NN	= "ATY,DV"	
	V	"ERE "	/S(NA)

```
             NN      = NN ",IBSLCE"
             V       *AAA/"25"* "1"         /F(NN42)
             NN      = NN ",S-1"            /(NN62)
RN           NN      = "AATY,DV"
             V       *SS/"34"* "1"          /F(NS)
             NN      = NN ",OSLV"
             V       *BBB/"36"* "1"         /F(RN11)
             NN      = NN ",IAPLCI"         /(SN)
RN11         V       BBB "2"                /F(RN12)
             NN      = NN ",IAPLVVI"        /(SN)
RN12         V       *CCC/"35"* "1"         /F(NS)
             NN      = NN ",CPLVO"          /(SN)
RN2          NN      = "AOTY,DV"            /(FS)

LN           V       "LE "                  /F(DN)
             V       *HH/"28"* "1"          /S(NN4)
             V       HH "2"                 /S(LN1)
             NN      = "AATY,DV,CSFCS"
             V       *SS/"34"* "1"          /F(NS)
             NN      = "AATY,DV,OSLCV"      /(NS)
LF1          V       "ELE "                 /F(RN2)
             NN      = "AOTY,DV,CSLVO"      /(NS)
```

DN	V	"DE "	/F(TN)
	V	*HH/"23"* "2"	/S(DN2)
	V	HH "3"	/S(NN1)
	V	HH "4"	/S(SHN0)
	V	HH "5"	/S(DN0)
	V	HH "6"	/S(DN3)
	V	*DDD/"29"* "1"	/F(DN1)
DN0	NN	= "ATY,DCV"	/(NS)
DN1	NN	= "ASTY,DCV"	
DN11	V	*EEE/"31"* "3"	/S(FA1)F(NS)
DN2	NN	= "AISTY,DVCV"	/(NS)
DN3	NN	= "AITY,DV"	/(NS)
TN	V	"TE "	/S(DN1)
VN	V	*HH/"28"* "1"	/S(RN1)
	V	HH "2"	/S(DN3)
	V	HH "3"	/S(VN1)
VN0	NN	= "ATY,DV"	/(DN11)
VN1	V	*DDD/"29"* "1"	/S(DN1)
	NN	= "ATY,DCV"	/(NA)
JN	V	"JE "	/F(IN)
	V	"LJE "	/S(LJN)

	V	"*JE "	/S(SJN)
	V	"UJE "	/S(UJN)
	V	*HH/"28"* "1"	/S(JN2)
	V	*AAA/"25"* "1"	/S(JN1)
	V	*DDD/"29"* "1"	/F(VN0)
	NN	= "ASTY,DV"	/(DN11)
JN1	V	"AJE "	/F(VN0)
JN1	NN	= "AVATY,DV"	/(SN)
LJN	NN	= "AYTY,DCV"	
LJN1	V	*FFF/"34"* "1"	/S(NN421)
	V	FFF "2"	/F(NS)
	NN	= NN ",OPFVPLV"	/(SN)
SJN	NN	= "AYTY,D*V"	/(LJN1)
UJN	V	*HH/"28"* "1"	/S(VN0)
	V	*GGG/"25"* "2"	/F(JN11)
	NN	= "AVATY,DV,S1"	/(SN)
JN2	NN	= "AJATY,DV"	/(NS)
IN	V	"JITJ "	/F(CN)
	V	*HH/"28"* "1"	/F(NTV1)
	NN	= "AJATY,DVC"	/(NS)
CN	V	"ASTJ "	/S(DAN)
	NN	= "ATY,DC"	
	V	*HHH/"32"* "1"	/S(NS1)F(SN)
DAN	NN	= "ATY,DCC"	/(SN)

```
FINAL   FVB     =
        CHGR    = "WRPL"
        TVB     = NVB
RTST    TVB     *R2* "+" =                          /F(NOR2)
LSTR    R2      = "+" R2 ","
        R2      "+A" *R22* ","                      /F(NOA)
        R22     = "+A" R22
DLAD    FVB     = FVB R22
        R2      R22 =                               /(YESA)
NOA     R2      "+D" *R23* ","                      /F(GETS)
        R22     = "+D" R23                          /(DLAD)
YESA    R2      ",D" *R24* ","                      /F(GETS)
        R22     = ",D" R24                          /(DLAD)
GETS    R2      ",S" *R25* ","                      /F(NOS)
        R22     = ",S" R25                          /(DLAD)
NOS     R2      ",I" *R26* ","                      /F(NOI)
        R22     = ",I" R26                          /(DLAD)
NOI     R2      ",O" *R27* ","                      /F(NOO)
        R22     = ",O" R27                          /(DLAD)
NOO     R2      ",C" *R28* ","                      /F($CHGR)
        R22     = ",C" R28                          /(DLAD)
WRPL    FVB     = FVB "+"
        R2      =                                   /(RTST)
NOR2    CHGR    = "WRPR"
        TVB     *R2* "," =                          /S(LSTR)
WRPR    FVB     = FVB "."
        READ    ("VTPS","4")
        PRINT     ("RTG","1")
RDVTP   VTPS      *CMVB* "." *NMR*                          /F(NOVT)
        CMPR      = CMVB "."
        EQUALS    (FVB,CMPR)                                /S(EQTST)F(RDVTP)
EQTST   NMR       "1"                                       /S(ONWR)
        V         = V ".0.0"                                /(STVERB)
ONWR    V         = V ".0.1"                                /(STVERB)
NOVT    V         = V ".1.0"
STVERB  RTG       = V                                       /(READIN)
END     BEGIN
```

Appendix 8

```
BEGIN    READ    ("VTP","3")
         PRINT   ("USNV","4")
RDVB1    VTP     *V* "." *R*                                              /F(USEIT1)
         CHR     = "RDVB1"
         SHR     = "AFFR"
SPLIT    R       *A* "." *B* "." *C* "." *D* "." *E* "." *F* "." *G* "." *H*
GH       H       = "." H "+"
         H       ".0+"                                                    /F(NZ)
         HR      = "Z"                                                    /(GG)
NZ       H       ".1+"                                                    /F(NO)
         HR      = "O"                                                    /(GG)
NO       H       ".2+"                                                    /F($CHR)
         HR      = "T"
GG       G       = "." G "+"
         G       ".0+"                                                    /F(RZ)
         GR      = "Z"                                                    /(GT)
RZ       G       ".1+"                                                    /F(RO)
         GR      = "O"                                                    /(GT)
RO       G       ".2+"                                                    /F(RM)
         GR      = "T"                                                    /(GT)
RM       GR      = "M"
GT       TR      = HR GR                                                  /($SHR)
AFFR     FA      = A
         FB      = B
         FVR     = "+" TR "+"
         FV      = V "." R
         FVR     "OM"                                                     /S(USEIT1)
```

```
RDVB2   VTP      *V* "." *R*                       /F(USEIT1)
        CHR      = "RDVB2"
        SHR      = "AFSC"                          /(SPLIT)
AFSC    SA       = A
        SB       = B
        SVR      = "+" TR "+"
        SV       = V "." R
        SVR      "OM"                              /S(USEIT2)
        EQUALS   (FVR,SVR)                         /S(TSTVT)
        SVR      "+O"                              /S(SVO)
        FVR      "+O"                              /S(RDVB2)
        SVR      "+Z"                              /S(SVZ)
        FVR      "+Z"                              /S(FVZ)
GTSCHF  SVR      "M+"                              /S(RPLFV)
        FVR      "M+"                              /S(RDVB2)
        SVR      "T+"                              /S(RPLFV)
        FVR      "T+"                              /S(RDVB2)
        SVR      "O+"                              /S(RPLFV)F(RDVB2)
TSTVT   FA       SA                                /F(TSTNS)
        F2       SB                                /F(TSTRS)
TSTNS   FVR      "+Z"                              /F(FNOZ)
        FA       "+1"                              /S(RDVB2)F(RPLFV)
FNOZ    FA       "+1"                              /S(RPLFV)F(RDVB2)
TSTRS   FB       "+1"                              /S(RDVB2)F(RPLFV)
SVO     FVR      "+O"                              /S(GTSCHF)F(RPLFV)
SVZ     FVR      "+Z"                              /S(GTSCHF)
```

```
        ZGT     = "RPLFV"
        NZGT    = "RDVB2"
SVZGT   SVR     "M+"                                /S($ZGT)
        SVR     "T+"                                /S($ZGT)
        SVR     "O+"                                /S($ZGT)F($NZGT)
FVZ     ZGT     = "RDVB2"
        NZGT    = "RPLFV"                           /(SVZGT)
RPLFV   FV      = SV
        FA      = SA
        FB      = SB
        FVR     = SVR                               /(RDVB2)
USEIT1  USNV    = FV                                /(END)
USEIT2  USNV    = SV
END     BEGIN
```

Appendix 9

```
BEGIN    PRINT    ("SAME","2")
RDUK     SYSPIT   *UL* "=" *US* ","                          /F(END)
         READ     ("ECN","1")
GTENC    ECN      *EL* "=" *ES* ","                          /F(PRDFF)
         READ     ("SLT","3")
GTTSTL   SLT      *UTL* " " *X* "," *ETL* " " *Y*            /F(RWLT)
         EQUALS   (UL,UTL)                                   /F(GTTSTL)
         EQUALS   (EL,ETL)                                   /F(RWLT)
         READ     ("SSD","4")
GTTSTS   SSD      *UTS* "," *ETS*                            /F(RWSD)
         UTS      *A* " "                                    /F(NOUB)
         UTS      = A
NOUB     ETS      *B* " "                                    /F(NOEB)
         ETS      = B
NOEB     EQUALS   (US,UTS)                                   /F(GTTSTS)
         EQUALS   (ES,ETS)                                   /F(RWSD)
         IDCN     = EL "=" ES ":" UL "=" US "."
         SAME     = IDCN                                     /(RWEC)
RWEC     REWIND   (ECN)
         REWIND   (SSD)
         REWIND   (SLT)                                      /(RDUK)
RWSD     REWIND   (SSD)
         REWIND   (SLT)                                      /(GTECN)
RWLT     REWIND   (SLT)                                      /(GTECN)
PRDFF    NEW      = UL "=" US ","
         SYSPOT   = NEW                                      /(RWEC)
END      BEGIN
```

Appendix 10

```
BEGIN     READ      ("NUMRS","1")
          AFRN      = "BRDF"
RDN       NUMRS     *N*
          N         *F/"8"* *R*                              /($AFRN)
BRDF      AFRN      = "TFBK"
RDF       SYSPIT    *A*                                      /F(END)
          A         *B/"4"* *C/"2"* *D/"4"* *E/"25"* *G/"47"*
TFBK      F         "BLK"                                    /S(STTST)
          F         *F1/"6"* *HF*
          C         = B C
          S         = F1 - C
          S         "-"                                      /S(FOULUP)
STTST     EQUALS    (F1,C)                                   /S(CHG)
NSTST     F2        "++"                                     /S(WTIT1)
          E         *P/F2* *Q*
          NE        = P "+" Q                                /(WTIT2)
WTIT1     SYSPOT    = B C D "+" E G                          /(FGBK)
WTIT2     SYSPOT    = B C D NE G                             /(FGBK)
CHG       F2        = HF
          R         *F1/"2"* *HF/"2"* =                      /S(NSTST)F(RDN)
FGBK      F         = "BLK"
          F1        *K/"4"*                                  /F(RDF)
          F1        K =                                      /(RDF)
FOULUP    SYSPOT    = "OUT OF PHASE."
END       BEGIN
```